Tailored random graph ensembles

Roberts, Ekaterina Sergeevna

Awarding institution:
King's College London

Thesis submitted to King's College London for the degree of Doctor of Philosophy

TAILORED RANDOM GRAPH ENSEMBLES

July 16, 2014

Candidate: Ekaterina S. Roberts

Supervised by: Professor Anthonius Coolen

Professor Franca Fraternali

To James.

Always.

ABSTRACT

Tailored graph ensembles are a developing bridge between statistical mechanics and biological networks. In this thesis, this concept is used to generate a suite of rigorous mathematical tools to quantify and compare the topology of cellular signalling networks.

Earlier published results to quantify the entropy of constrained random graph ensembles are extended by looking at constraints relating to directed graphs, bipartite graphs, neighbourhood compositions and generalised degrees. To incorporate constraints relating to the average number of short loops, a number of innovative techniques are reviewed and extended, moving the analysis beyond the usual tree-like assumption. The generation of unbiased sample networks under some of these new constraints is studied.

A series of illustrations of how these concepts may be applied to systems biology are developed. Topological observables are obtained from real biological networks and the entropy of the associated random graph ensemble is calculated. Certain studies on over-represented motifs are replicated and the influence of the choice of null model is considered. The correlation between the topological role of each protein and its lethality is studied in yeast.

Throughout, this document aims to promote looking at a network as a realisation satisfying certain constraints rather than just as a list of nodes and edges. This may initially seem to be an abstract approach, but it is in fact a more natural viewpoint within which to consider many fundamental questions regarding the origin, function and design of observed real networks.

ACKNOWLEDGEMENTS

To Ton and Franca, who showed incredible faith to even let me take the first step in this journey - and unstinting generous support every step of the way. To my Mum, for her boundless generous enthusiasm for the little people: their adventures, their education and a little light conversation. To my Dad, whose studied indifference and constructive disapproval belies the fact that - over a decade after I officially left - if he is not underwriting our homestead, he's surrendering his house to a nursery. To Lesley who has seen our family from scrawny infants, through high-emotion-low-articulation toddlers to the young people they are today, always with love and good spirits. We couldn't have done it without you (and we miss you!).

To Xenia, my little princess growing up into a lovely young woman, whose imagination has brightened up many a long day, and whose kindness set our growing family the best possible example. The cat who walks by herself - I see myself reflected in your eyes. Walk safe and walk brave, little one. To Peter, my big hearted pirate, with his attentive, careful ways balanced with his enthusiasm for whatever adventure we can draw him into. And who likes my food nearly always - so is my big hope for a well-dined old age. May there be many more stories ahead for you. To Leonard, with the sparkling eyes. This thesis has been four years in the writing - and you haven't changed at all. I hope that you grow up to find a world as big and as loving as your heart. To Marina, my crocodile. So new and yet so familiar - it's like there was always an empty chair before you joined us - and now you're here, bringing sunshine.

To those who taught me and those who travelled with me. The lessons I learnt and the friendships I made continue to mould me. And finally, to the kindness of strangers, who grow to become friends. Those who I have mentioned, and also Roy, Valerie, Janet, and the late Ken Bullough and Mother Thekla. I admire your courage to let us into your lives, and I thank you for the opportunities you opened to me. And, of course, to James - who has been the rock on which I can build and the heart that makes it all matter. My mistakes are my own, but my successes belong to all of you.

PUBLICATIONS

E.S Roberts, T. Schlitt and A. C Coolen. Tailored graph ensembles as proxies or null models for real networks II: results on directed graphs. *Journal of Physics A Mathematical General*, 44(26):5002, 2011.

E.S Roberts and A. C Coolen. Unbiased degree-preserving randomization of directed binary networks. *Physical Review E*, 85:046103, 2012

E.S Roberts, A. Annibale and A. C Coolen. Tailored random graph ensembles. *Journal of Physics: Conference Series*, 410(1):012097, 2013

E.S Roberts, A. Annibale and A. C Coolen. Controlled Markovian dynamics of graphs: unbiased generation of random graphs with prescribed topological properties. In C. Gracio, editor, *Nonlinear Maps and their Applications* volume 57 of *Springer Proceedings in Mathematics and Statistics*, pages 25-34, 2014

E.S Roberts and A. C Coolen. Entropies of tailored random graph ensembles: bipartite graphs, generalised degrees, and node neighbourhoods. Submitted to: *Journal of Physics A Mathematical General* 2014

E.S Roberts and A. C Coolen. Random graph ensembles with many short loops. Submitted to: *ESAIM: Proceedings* 2014

AFFILIATION

This work has been hosted by the following research groups:

Institute for Mathematical and Molecular Biomedicine, King's College London

Randall Division of Cell and Molecular Biophysics, King's College London (Fraternali Group)

Department of Mathematics, King's College London (Disordered Systems Group)

Genetics Department, King's College London (Schlitt Group)

This work was funded by the Biotechnology and Biological Sciences Research Council.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND NOTATION

Symbols

S	Shannon entropy, see equation (2.3.1), page 36
N	Number of nodes, page 32
k	Degree (number of neighbours), page 32
$\vec{k}$	Directed degree, page 48
$\boldsymbol{k}$	Boldface letters represent ordered sets with N elements, i.e. $(k_1, \ldots, k_N)$, page 47
$\bar{k}$	Average degree, page 32
$p(k)$	Degree distribution, page 32
$\langle f(\boldsymbol{c})\rangle_{\bar{k}}$	Average of $f(\boldsymbol{c})$ over Erdös-Rényi ensemble with average degree $\bar{k}$, page 49
C	Complexity, page 56
$W(\cdot,\cdot)$	Joint distribution connected degrees, page 171
D_{AB}	Kullback-Leibler distance between two ensembles, see equation (3.3.1), page 54
$\mathcal{N}$	Effective number of graphs, page 48
ϵ_N, ε_N	Corrections that vanish in the large N limit

f For bipartite networks, giving the probability that a node is in set A, page 65

(k_i, m_i) Generalised degree of a node i, giving number of neighbours within one and two steps respectively, page 60

H Hamiltonian, see equation (2.3.2), page 36

$n_i = (k_i;\ \{\xi_i^s\})$ Local neighbourhood of a node i, see equation (4.1.1), page 60

$w(\boldsymbol{c}|\bar{k})$ Poissonian measure, see equation (3.1.4), page 49

$n_\square$,$n_\triangle$ Mobility factors, see equation (6.1.8), page 101

$\{\xi_i^s\}$ Degrees of neighbours of i, page 60

$O()$ 'Order of', referring to the magnitude of residual terms in a leading order calculation

Z Partition function, see equation (2.3.3), page 36

$\pi_{\bar{k}}(k)$ Poissonian distribution , page 50

$p_\infty(\boldsymbol{c})$ Equilibrium measure, page 99

$Q(,)$ Kernel to impose specific correlation profile, see equation (3.2.4), page 52

$Tr(\boldsymbol{c})$ Sum of diagonal entries of the matrix $\boldsymbol{c}$.

Z_{ER} Partition function of Erdös-Rényi ensemble, see equation (5.2.5), page 78

Terminology

Bipartite graph: a graph where the nodes can be divided into two sets such that links only appear between sets, and not within sets, page 60

Edge swap: degree preserving randomising move, page 97

Ensemble: a collection of systems defined via some constraint and/or probability measure

Ergodic: a process which can move between any two states in a finite number of steps , page 41

Eukaryote: organism whose cells have a distinct membrane bound nucleus, e.g. mammals, page 34

Hub: node with degree greater than 20, page 146

Law of large numbers: formal statement of the intuitive concept that the average over a large number of observations should be close to the expected average, and the agreement should improve the more trials are performed

Motif: small (over-represented) subgraphs, page 135

PPIN: protein-protein interaction network, page 32

Prokaryote: organisms without a membrane-bound nucleus, typically single celled, such as bacteria

ROC: plot of true positive versus false positives in order to evaluate model performance

Sub-hub: node with 5 <degree≤ 20, page 146

E. coli: a bacteria, and a widely studied model organism, page 128

H. sapiens: Homo sapiens (humans), page 128

P. faciparum: parasite implicated in malaria, page 128

S. cerevisiae: yeast. Popular eukaryotic model organism, page 128

T. pallidum: a bacteria, page 128

CHAPTER 1

INTRODUCTION

Networks are an end point of simplifying a complex system to its simplest meaningful representation. They are familiar to every person, generally as some kind of map. Whether it is finding a convenient route on public transport or connecting the wiring in a home improvement project, a complex system can be easier to understand if it is represented in terms of its components (nodes) and inter-relationships between those components (links). Human beings are skilled at observing patterns in networks; shortcuts, motifs and redundancies become evident when the system's interactions are clearly charted. However, as the network's size increases, intuition can become a false-friend, since it is difficult to distinguish meaningful patterns from random fluctuations. This thesis will be specifically interested in developing rigorous mathematical tools to quantify and analyse the topology of networks based on biological datasets.

The current state of the art of biological research is now approaching some of the most detailed questions about the self-organisation of components that constitutes what we know as life. Molecular biology is a present day space race. It coincides truly fundamental scientific questions with transformative technological advances. Like the space race, success will only be possible with cross-disciplinary collaboration, and the development of innovative new tools and approaches. There is now a largely complete list of the components (see e.g. descriptions in Schlitt and Brazma [2007]) of the key model organisms. The most famous contributor is the

Human Genome Project - but that of course represents only a small part of the total achievements in this field. The advent of high-throughput experimental techniques means that much more is now known about the interactions and inter-relationships between the components. The volume of available data is now vast, and analysing it efficiently and rigorously is vital for the integrity of the whole endeavour. In the physical sciences, statistical mechanics specialises in the analysis of large systems, and has a heritage of rigorous and practical techniques. However, it would be arrogant and ineffective to presume that these results could be immediately translated to the very different context of living organisms. It is important to take some steps back, and to systematically re-tailor traditional statistical mechanics results for application to biology. The approach of statistical mechanics is to define large systems through some simple rules and assumptions. The simplifying instincts of physicists can seem obtuse and self-serving to an experienced biologist, whereas a physicist may be frustrated by the impossibility of adding rigour to a model that is bespoke to the point of being descriptive. Despite certain universalities, biological systems are generally interesting for their differences. Equally, the complexity of contemporary biological data means that certain simplifying rules of thumb have enjoyed great success, despite being frequently challenged as not adequately capturing the features of the system (Barabási and Albert [1999], Erdös and Rényi [1960]). In this thesis we will aim to increase the flexibility of certain techniques in statistical mechanics, in order to allow scope for the models to be tailored to biological knowledge and insight.

One line of research is the growing network approach (e.g. Albert and Barabási [2002]) which typically starts with a single node and additional nodes are added, and connected to the pre-existing nodes according to some rules. In the ideal implementation, the rules would reflect some universalities or constraints that are believed to hold for networks of that type. The final network is then a more finely tuned null-model, to be used either as a testing ground or for comparisons with the original network. While the growing network approaches are intuitively attractive, they suffer from not having clearly defined probabilities associated with each final state. So there is no guarantee that the observations are not a result of inadvertent model bias.

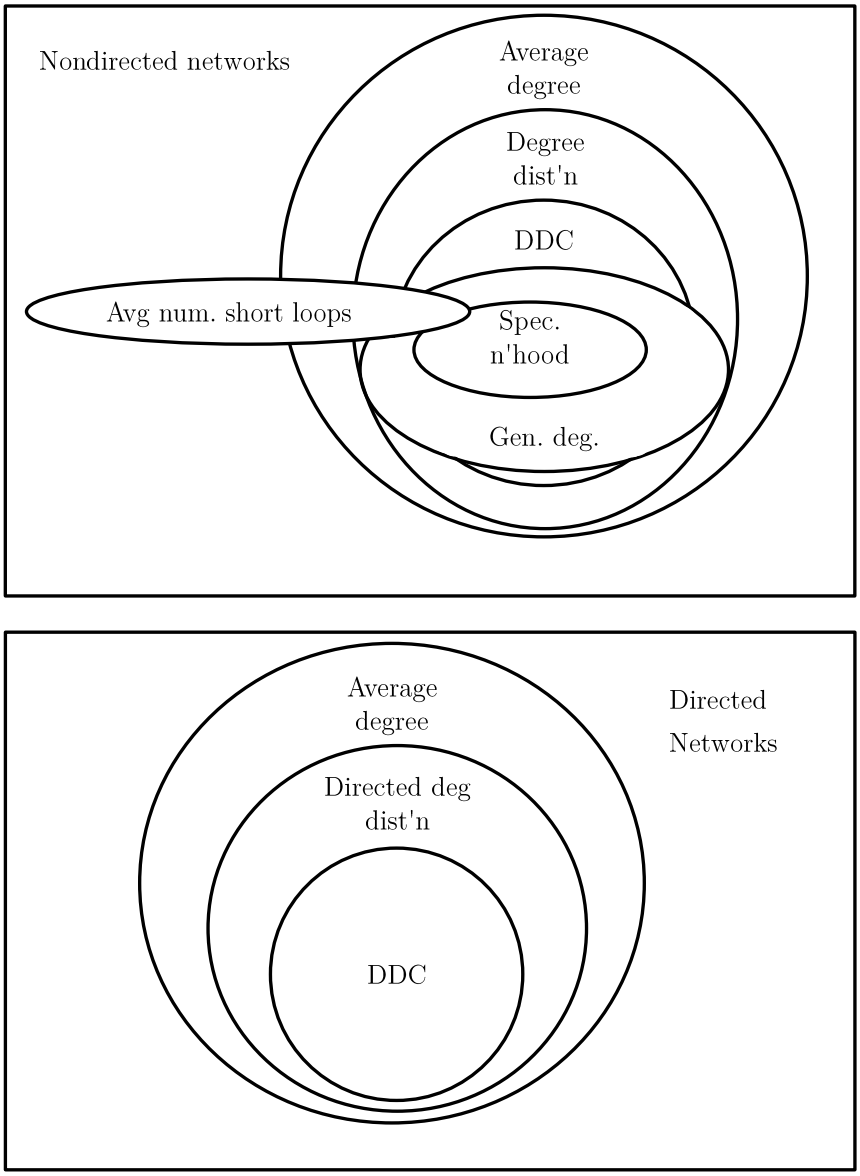


Figure 1.1: Venn diagram of various constrained ensembles, the size of which will be quantified in the course of this thesis. Illustrations not to scale, but indicate the overlaps between the different definitions. Abbreviations refer to degree-degree correlation, degree distribution, specified neighbourhoods and generalised degrees.

There are contrasting routes to try to connect networks, biology and mathematics into a meaningful and useful theoretical framework. Many authors have sought to find universal topological patterns. For example, it is often taken as a fact that biological networks have a power law distribution in the number of neighbours that each node has, even though several authors have challenged this as a useful shorthand but not an accurate description. As a bridge to traditionally local biological approaches, it has been popular to seek over-represented small sub-graphs (motifs) in biological networks, because this then opens up opportunities for theories to be developed about the biological processes that may underlie these motifs. The concept of a random graph has appeared typically as a null-model, in order to provide the control case to give statistical significance and context to what is observed in the real network.

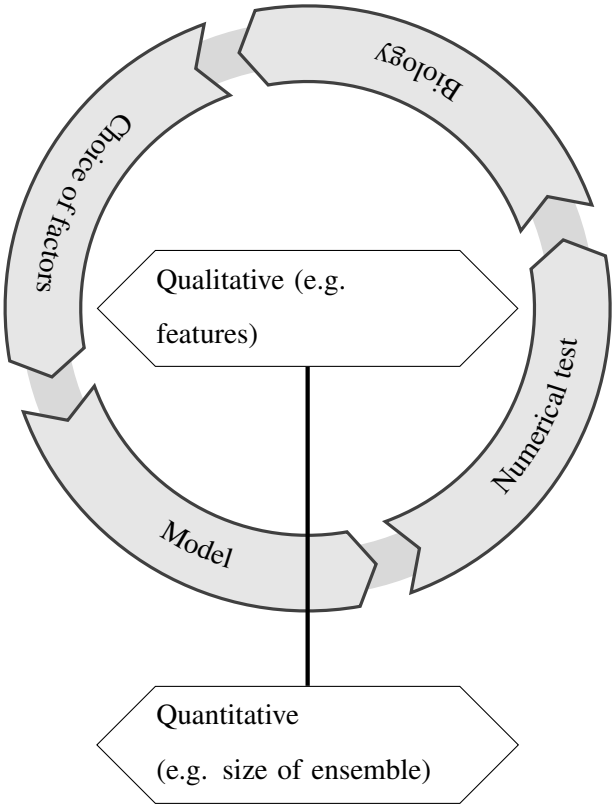


Figure 1.2: Refinement of biological models, through numerical tests and analytical challenge - e.g. whether the qualitative features of the model comply with what is needed, and how restrictive the constraints are.

The work in this thesis looks at a network as a realization satisfying certain specified topological properties rather that simply being a set of linked nodes. This is a natural viewpoint if we were to go on to — for example — consider what underlying processes formed the network or what function it is 'designed' to carry out. It enables a more rigorous interrogation of the network: What are its features? Is it typical of a network with such features? What functions or processes would be consistent with this topology? Which synthetic model is appropriate and sufficiently close to be useful as an analytically tractable proxy for the real network? Asking any researcher to describe what makes a network complex, the reply may refer to degree distribution, clustering coefficient, communities or heterogeneity of nodes. There is a large literature describing these features, some of which is briefly covered in chapter 2. In this thesis we will formalise this idea by quantifying topological observables through information theory.

The purpose of this thesis is to extend the understanding of ensembles of random networks which are defined with topological constraints. This is approached from two directions. Firstly, the tools of statistical mechanics are applied to quantify the entropy of such ensembles under various constraints. This is split across two parts. Chapters 3 and 4 present new results for the entropy of ensembles constrained with in- and out- degrees (directed) and undirected ensembles constrained with a specified generalised degree distribution respectively. Chapter 5 works towards determining the entropy of ensembles where the constraint is specified in terms of the distribution of short loops in the network. These constraints are interrelated - the Venn diagram in figure 1.1 gives a rough illustration of the hierarchy. The approach used in chapter 3 fails if the networks are no longer tree like - so is not applicable to short loop constraints. Hence chapter 5 explores a selection of mathematical topics which offer potential inroads and insights into the problem. Chapter 6 studies unbiased degree preserving randomisation of directed binary networks. This organisational structure is summarised in figure 1.3. The main new results are:

- Quantifying the entropy of random graph ensembles under new constraints
  - Exactly in the large N limit for
    - Ensembles of directed networks constrained with degree distribution and degree-degree correlation
    - Ensembles of bipartite networks constrained with degree distribution
    - Ensembles of non-directed networks constrained with specified neighbourhood distribution or specified generalised degrees

  - Estimated or partial solutions for
    * The maximum entropy ensemble for a given average number of short loops
    * The monopartite projection of a bipartite network, which is motivated by analogy with protein complexes

- Novel or improved linkages between the concept of a constrained random graph ensemble and the needs of bioinformatics and molecular biology, including:
  - New results on how to construct unbiased constrained randomisation algorithms for directed graphs
  - Novel illustrations of the influence of the choice of null-model on the study of over-represented motifs
  - An expansion of the idea of being able to quantify topological features, by cross-referencing with lethality data, in order to consider whether topology is predictive of whether the node plays an essential role in the network

The various research areas of bio-informatics are often illustrated as a hierarchy from the most granular level of looking at protein binding domains, to the intermediate level of pathway analysis which looks at disease of function specific (sub)networks, culminating in the most global and large scale view which attempts to catalogue the interaction between every gene or protein within the system. The benefit of the global view is that it summarises and aggregates all of the sub-systems that make up, and interact within, the organism. The first difficulty with the global view is the reliability of the data. The large automated component of either high-throughput screening or data mining mean that many nuances may go undetected. Secondly, it is not clear how to interpret this data, even in simplified form, let alone incorporating the complexities of biological annotations, strength of interaction, localisation of interaction and so on. The path to leveraging local knowledge in order to form a systems wide view is much better established than the reverse route: using global information to discover new granular results. This observation is not intended to detract from some encouraging systems biology successes (e.g. the work of Chuang et al. [2007] on network based classification of breast cancer metastasis). However, in the way that genes are the language that allows us to 'read' inheritance, and a network is a way to 'read' large volume interaction data - it is necessary to discover and develop a natural language to read and interpret a network.

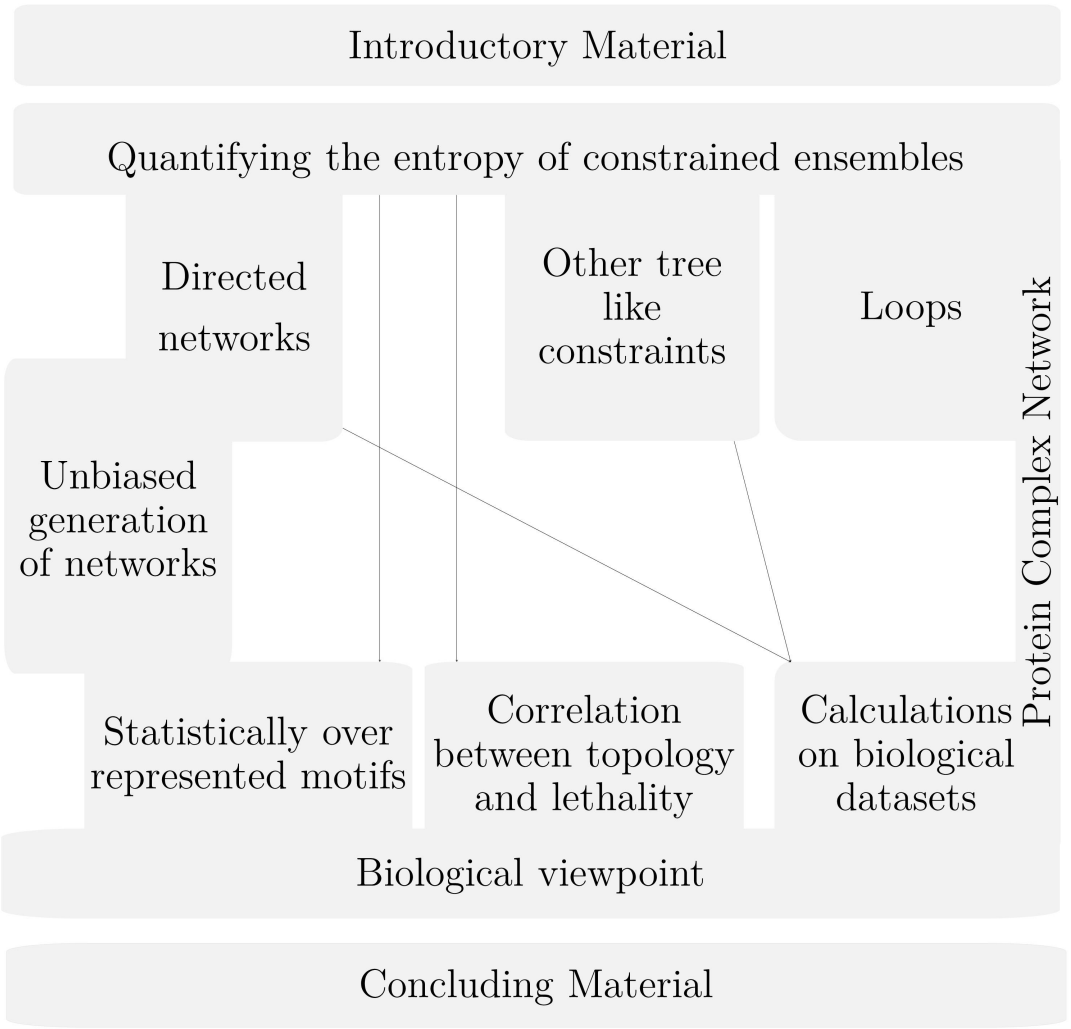


Figure 1.3: Summary of the organisation of the thesis. Arrows indicate connections explicitly made, although in principle any item on the mid-top line can be constructively related to any item on the mid-bottom line.

The normal workflow to modelling a real system (illustrated in figure 1.2) would be to select some key features or parameters, incorporate them into a model and then test the predictions of the model - repeating the process until the model is sufficiently predictive for the required application. In the language of this thesis and the language of network science the first part of this process would be to use biological intuition to select which features of the network are considered to be the most important, and then to define a model which captures those features. The cleanest approach is when the model is otherwise maximally random. Testing the predictive power of a model is not entirely straightforward, so it makes sense to apply rational steps to challenge proposed models at an early stage. In the illustration in figure 1.2 this is broken down as a qualitative and a quantitative challenge. Qualitatively, there are many interesting results in the literature, which look at typical topological features that follow as a consequence of imposing a constraint on the random graph ensemble. For example, Molloy and Reed [1998] discovered that a giant component would almost certainly appear if the imposed degree distribution $p(k)$ satisfies $\sum_k k(k-1)p(k) > 0$. Chung and Lu [2004] studied shortest path length for networks with a given degree sequence and found it to be almost surely of the order $\frac{\log N}{\log\langle k^2\rangle}$, where the quotient is the average value of the squared degree. Chapters 3, 4 and 5 of this thesis aim to provide a route for providing a quantitative challenge and comparison for network topology based models, in the sense of aiming to quantify the constraints by how restrictive they are (and hence how specific the proposed model is). The remaining material looks at the other parts of the cycle set out in figure 1.2.

CHAPTER 2

LITERATURE AND PRELIMINARIES

This chapter will provide a general overview of the theoretical background and underpinning to the results which follow. For clarity, more specific literature is reviewed briefly at the beginning of each chapter. The purpose of this chapter is to elucidate the relationship between networks, molecular cell biology and statistical mechanics that is the conceptual starting point of this thesis. These are separately well established and well developed fields of study. The study of networks is traditionally considered to date back to Euler's study of the bridges of Königsberg (reproduced in Euler [1953]), where the crucial observation was that the topology of the problem was more important than the geometry. Molecular cell biology is an exciting and fast developing field, driven by rapidly improving experimental techniques, and ambitious scientific objectives — but challenged by the complexity of living organisms. Statistical mechanics dates back to the work of Maxwell, Boltzmann and Gibbs in the 19th century. Classic statistical mechanics problems would look to derive the macroscopic properties of a gas via the microscopic behaviour of gas molecules, or the macroscopic property of magnetisation via microscopic nearest neighbour interactions in a metallic lattice. In this section we will separately explain some key pre-requisites in these fields, and also aim to give a flavour of how networks can help interpret complex data, and how statistical mechanics can be employed to rigorously derive some average properties of large model systems or networks. We will also cover some

tangential examples of the application of information theory to biological problems, in order to provide some context for later work.

Overall, these fields of study are very large, with new important publications regularly appearing. This chapter tries to limit itself to those areas of the literature that would particularly contribute to the understanding of what follows. This means that several interesting topics are outside of our scope. A wider view of systems biology, particularly as it relates to networks, can be obtained from books such as Newman [2009], Alon [2006], Dorogovtsev and Mendes [2002] and review articles including Dorogovtsev et al. [2008], Albert and Barabási [2002].

2.1 Networks

Networks are interesting in themselves as objects of mathematical study. However, they particularly attract attention because of their power to visually and informatively represent complicated data sets. Common applications include modelling the internet (see e.g. Zhou and Mondragón [2004]), modelling social networks (see e.g. Wasserman [1994]), communications networks (see e.g. Uddin et al. [2011]) and power grid networks (see e.g. Pagani and Aiello [2013]). We will particularly be interested in applications to biology, where we are now able to harvest large amounts of information and hope to be able to infer some global insights about the system. We will specifically be looking at gene-regulation networks and protein-protein interaction networks, but the flexibility of the networks concept means that it appears in a very wide range of other biological contexts, including metabolic networks and food chains. The spirit of mathematics is to look for general results that can then be re-used for different problems which have the same underlying structure and symmetries. In this way, it makes sense to study networks abstractly, with the intention of applying the results to a certain kind of network, but with the prospect that the tools will be relevant for any kind of network.

A network (which will interchangeably be referred to as a graph) is defined via a set of N nodes, and a set of edges between pairs of nodes. The network may be defined as a projection of a dataset of binary interactions (e.g. telephone calls between people), or it may be defined via a model. Figure 2.1 shows an example of a random network, and demonstrates how much the visual appearance of a network can be influenced by the layout algorithm chosen. We will typically be working with the simplest case, where all the nodes are assumed to be identical,

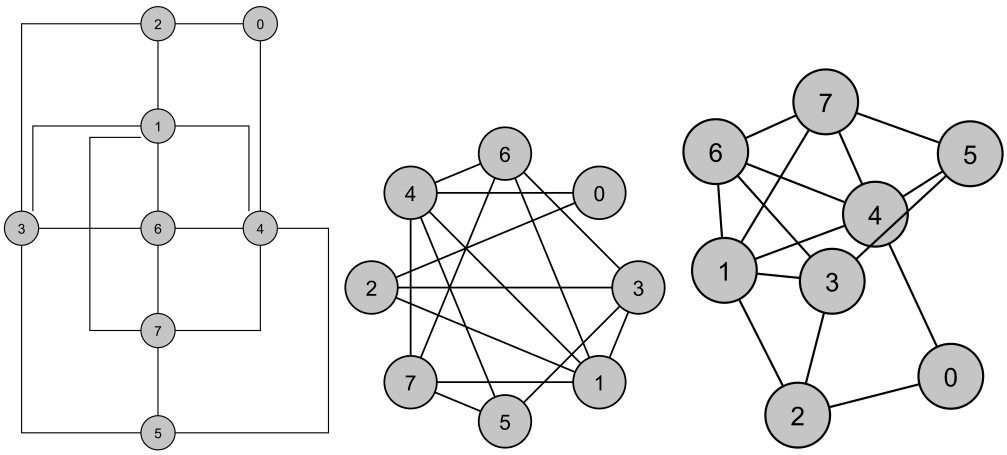


Figure 2.1: A network is defined as a set of nodes and connections between them. Visualising a network can help to understand its structure. However, the same network can look very different depending on the visualisation algorithm used. The above illustration shows the same random network with three different layouts.

and double or self links are forbidden. To make a connection with statistical mechanics, the networks have to be large networks, often idealised to be infinitely large with $N \to \infty$. In practice, sizes of several thousand nodes are sufficiently large to apply derived tools, which is commensurate with the typical sizes of protein-protein interaction databases.

Notation is summarised at the front of the thesis, and explained again when it occurs in the body of the thesis. A regular network has the same number of neighbours for every node. The Bethe lattice — a tree like network where every node has the same degree — is an example of an infinite regular network. A random graph ensemble is a family of networks which meet some (hard or soft) constraints, and the random network is chosen from such an ensemble. The most well known is the Erdös-Rényi random graph (Erdös and Rényi [1960]), where every edge is formed with a probability $p \in [0, 1]$ independently of any other edge. Its simplicity and analytical tractability has made it the canonical random graph model. However, it does not match many of the topological properties observed in real networks.

The preferential-attachment algorithm (Barabási and Albert [1999]) was designed as an explanation and a model for the 'fat tails' observed in real degree distributions (i.e. many more nodes with very high numbers of neighbours than would be predicted by the Erdös-Rényi model). New edges are added one by one, with a probability that depends on how many neighbours the vertices already have. This 'rich get richer' algorithm is the most well known case of growing

network models, which have had some success in describing the intuition and the features behind — for example— the Internet network. The Watts and Strogatz [1998] model is designed to show a higher degree of clustering (i.e. interconnected triples of nodes). It operates by rewiring a regular one-dimensional lattice. It demonstrates a 'small world' topology (i.e. short path length between any two nodes) which is similar to many real life networks. The latter two are examples of useful qualitative models which suffer from the drawback that the probability of each network in the ensemble is not clearly defined. In this thesis we are primarily interested in random graph ensembles which have a lot of flexibility to match parameters in real networks. These are specified by hard constraints (typically a fixed number of neighbours for every node), and by probability weights to add some soft constraints.

Tables 2.1 and 2.2 summarise some key network metrics, annotated with comments about how these metrics relate to the treatment in this thesis. Network science has grown extremely fast, driven by the urgent demand for effective tools to analyse the networks that arise in real life applications. Accessible treatments of the field can be read in Newman [2009] and Dorogovtsev and Mendes [2002]. The coverage of these books reflects the aspects of network science which have received the most attention in the literature: topological features, growth processes, vulnerabilities to attack.

Topological observable	Comments on usage in this thesis
Number of vertices N	These will typically be indexed as $i = 1 \dots N$. We will be interested in cases where N is large.
Degree k	Number of neighbours. In this work, k will be assumed to remain finite, even in very large networks
Degree distribution $p(k)$	Can be either specified, or obtained from a real network as $p(k) = \frac{1}{N} \sum_{i=1}^{N} \delta_{k,k_i}$
Average degree $\bar{k}$	We require this to stay finite as the network grows
Clique	Fully connected subset of nodes
Motif	Small subgraph appearing in the network
Assortativity	Tendency for nodes which are similar to be connected. Typically refers to tendency for high degree nodes to connect to each other

Table 2.1: An annotated list of some observables on a network that will be particularly referred to in this thesis.

2.2 Modern biology and the role of networks

Collecting biological data has traditionally been a slow and largely manual process. The advent of high throughput techniques has progressed the ambitions of practitioners from understanding discrete processes, to understanding biological systems as a whole. Below we review briefly the experimental techniques underlying the datasets, and some pertinent caveats to their interpretation. The exposition is mainly based on books by Lesk [2012, 2010], Fu [2004] and Newman [2009].

Topological Observable	Comments on usage in this thesis
Clustering	Ratio of fully interconnected triplets of nodes to connected triplets. We generally just look at the average number of closed triplets in the network $\bar{m}$
Shortest path/betweeness	These do not appear in this thesis. They are interesting properties, but are intrinsically global, so do not easily yield to the simplifications of statistical mechanics
Size of giant component	This does not directly appear in this thesis. The existence of the giant component will typically be a consequence of the parameters chosen for the random graph model
Hub	A node with many interacting partners

Table 2.2: An annotated list of additional observables on a network.

Co-immunoprecipitation is a reliable but laborious technique for detecting protein-protein interactions (PPI). A suitable antibody is attached to a glass column, and binds to the target protein. The target protein, and any protein bound to it, can be retrieved and analysed (e.g. by Western blot). The antibody has to be specific to the protein of interest. Although there are techniques to manufacture antibodies which do not naturally occur, it nonetheless requires one to *a priori* define and commit to at least one of the binding partners. Hence, this method is likely to be biased towards discovering interactions involving proteins which are already considered to be important — which will create a self-reinforcing loop of correlation between node degree and node significance.

Yeast two hybrid (Y2H) is a well-established high throughput method. The two proteins of interest are tagged with a DNA binding domain and a transcriptional activation domain. If these are brought into close proximity (i.e. if the two proteins to which they are attached interact), then this would trigger the production of a reporter gene. Its great strength is that it is possible to screen efficiently a single protein against a library of other proteins — thus finding interactions

independently of the prior views of the experimenter. It is important, however, to be alert to the high possibility of false positives, when the reported gene is activated without the proteins interacting, and also the possibility of false negatives. Best practice would be to confirm the interaction by another method — but this is limited by resource constraints. Generalisations of this technique include a membrane based approach — to address the fact that hydrophobic membrane proteins are difficult to study using existing techniques, but represent about a third of proteins in eukaryotes — a two-hybrid system based on *E. coli* and protein complementation assay.

The idea of tagging each protein so that a signal is emitted when they interact is also at the heart of the fluoresence resonance energy transfer (FRET) and fluoresence lifetime imaging microscopy (FLIM) methods. In this approach, Green Fluoresence Proteins (or variants thereof) are attached to the proteins of interest. Green light is emitted if the two tags are close enough together for energy transfer to take place. The likelihood of energy transfer varies like the sixth power of distance (Fu [2004]), making this an accurate quantitative tool. The major advantage of this method is that it can be used *in vivo*, with the light emitted being observed through a microscope.

Tandem Affinity Purification coupled with Mass Spectroscopy (TAP-MS) is a multistep procedure which has many of the advantages of the Y2H method and of co-immunoprecipitation. The antibody is linked to the target protein via an intermediate protein — which is known to interact with the antibody, and is thus attached to the target protein to act as a hook.

Alternative methods to construct protein-protein interaction networks include computational data mining (e.g. looking for protein names co-occuring in the abstracts of published papers), nuclear magnetic resonance (NMR) and phage display. It is a material issue that these varied experimental approached leave signature topological features — and it is not evident how to isolate these from the organism specific topological features (e.g. Fernandes et al. [2010]). Overall, this is a strong argument to concentrate efforts on the development of transferrable tools, recognising that the current realisations of biological networks will continue to be refined and clarified in years to come. The data is stored on public databases.

In chapter 3 we study directed networks. The natural application of these results are to gene regulation networks. Protein-protein interactions are symmetric, whereas gene regulation is not. The central dogma of molecular biology is that deoxyribonucleic acid (DNA) is transcribed to

'messenger' ribonucleic acid (mRNA), which encodes amino acid sequences, which are the building blocks of proteins. Some of these proteins belong to a class called 'transcriptional regulators', which are able to stimulate or inhibit expression of another gene. When we observe a gene regulation network, we are observing the action of the transcriptional regulators, and also the effect of the interactions between the transcriptional regulator proteins and non-regulating proteins. This is reflected in the topology of the gene regulation networks, where the out-degree distribution has a few hubs and many nodes with zero out-degree, while the in-degree distribution is more evenly distributed. This is an example of why Poissonian approximations are not adequate to capture essential features of the topology.

The question of comparing networks naturally arises, not least because there are now several significant PPI datasets for the model eukaryote and prokaryote organisms, and these themselves depend on the choices made for confidence thresholds and experimental interpretation. Comparison methods in the literature are typically of the algorithm type. For example, Kuchaiev et al. [2010] designed a comparison based on subgraph counts.

2.3 Statistical mechanics techniques and concepts used

In this section we will set out some preliminaries, in order to justify and define the general scheme to approach a 'solution' of the many models which are explored in this chapter and in this thesis. This exposition is based on the following textbooks: Bowley and Sanchez [1999] for an introduction to classical statistical mechanics; Feynman [1972] and Parisi [1998] for more in depth coverage; Dorogovtsev and Mendes [2002] for a more discursive coverage; and, Kardar [2013] lecture notes and Mézard et al. [1987] for an introduction to the replica method.

An ensemble is a collection of systems, generally defined via some constraint or property that the systems have in common and a probability measure. The term entropy appears separately in physics as Boltzmann entropy and thermodynamic entropy and in information theory as Shannon entropy. Shannon entropy is a concept that arose out of Claude Shannon's experience working on codes during the war. The reasoning behind Shannon's Information theory is roughly as follows. Suppose that you had to design a code to transmit words from a set A. If we were transmitting messages in binary code, a string of X digits (e.g. $01001\ldots1000$) could code 2^X distinct words. Hence a code to transmit words from set A must be based on at least

$\log_2 |A|$ binary digits (or bits). If we were transmitting words from 2 sets, A and B, we would need $\log_2 |A| + \log_2 |B|$ bits. If we also take into account the likelihood of every observation then this generalises to give the definition of the Shannon entropy for some random variable X with n outcomes $\{x_1 \dots x_n\}$

$$S = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{2.3.1}$$

This has been widely adopted as a tool with which to define measures of complexity (some examples are discussed in section 2.5) . The systems that we are interested in have non-trivial and non-constant probabilities of each state appearing. Using the language of statistical mechanics, we define the probability of state x, via a Hamiltonian H ('energy function'), as

$$p(x) = \frac{\mathrm{e}^{-H(x)}}{Z} \tag{2.3.2}$$

where Z is the normalising constant, usually referred to as a partition function

$$Z = \sum_{x} \mathrm{e}^{-H(x)} \tag{2.3.3}$$

Writing $\ln p(x) = -H(x) - \ln Z$ and substituting into the entropy equation allows $-k_B T \ln Z$ to be identified as the (experimentally measurable) Helmholtz free energy with k_B as the Boltzmann constant and T the temperature. In this thesis, for simplicity, we will continue to refer to $-\ln Z$ as the free energy, and will show in context the meaning of this quantity for our applications. The free energy is typically the key quantity in the problem, with many observables being directly derived from it.

Calculating the free energy is often a difficult problem. Baxter [2007] carefully sets out the strategies that may be pursued to try to evaluate this quantity. There are a number of models which can be solved exactly, but these are generally very idealised cases, lying at the extremes of either the almost trivial one-dimensional (i.e. nearest neighbour interactions on a line) cases, or infinite-dimensional cases. For more complicated cases — to paraphrase the view of Baxter [2007] — the options are to substitute a simpler Hamiltonian (i.e. solving a simpler model), and/or make an approximation to evaluate the sum over states. In relation to real systems, a well defined simple model can give valuable information about the qualitative behaviour of the system, particularly around any critical points. The second proposal laid out in Baxter [2007] is to find an approximation to do the sum over states, for example:

- Cell approximation (i.e. viewing a system as interacting neighbourhoods)

- Approximate integral equations
- Computer simulations
- Series expansions
- Renormalization group approach

Newer methods which are particularly designed to cope with heterogeneous systems include

- The cavity approach (see e.g. Mézard et al. [1987])
- The replica approach (see e.g. Mézard et al. [1987])

The material in this thesis will use many of these ideas — presenting solutions that aim to be exact in the limit by a suitable definition of the underlying model, and also adapting elements from series expansions, cell approximations, integral approximations, computer simulations and the replica approach, as the opportunity arises. Below we explore these specific approximations further. We begin with a definition of the Ising model, in order to provide a base for subsequent examples.

2.3.1 Ising model

The Ising [1925] model is the classic model for co-operative behaviour in large systems. It was originally developed to model magnetisation of iron. The vertices are arranged along a line or an n-dimensional lattice, and each vertex i carries a spin of $\sigma_i \in \{-1, 1\}$. If we set J to be the strength of the ferromagnetic interaction between nearest neighbour spins, and h as the magnetic field, then the Hamiltonian is defined as

$$H = -J \sum_{(i,j)} \sigma_i \sigma_j - h \sum_i \sigma_i \tag{2.3.4}$$

The sum is done over nearest neighbours on the lattice. The properties of the system depend on the sign of J ($J > 0$ is the ferromagnetic case). The magnitude of J determines the critical temperature where magnetism (i.e. long range order) is lost.

2.3.2 Series expansions

The principle of the series expansion approach is to expand the exponential inside the logarithm of equation 2.3.3 in order to obtain a power series, which can ultimately be truncated by applying some prior knowledge of how the parameters scale. This section is based on material from Kardar [2013] and Parisi [1998]. Series expansions are typically classified as either low-temperature or high-temperature expansions. The low-temperature expansion starts from the ground state, where all of the spins are set to have the same value. High temperature expansions begin with non-interacting spins at infinite temperature and then perturbatively include the interactions between spins. For a brief example of a low temperature expansion (i.e. when K is large), consider the Ising Model on a 2 dimensional lattice, without an external magnetic field. Define the probability of any configuration to be

$$p(\boldsymbol{\sigma}) = \frac{\mathrm{e}^{K \sum_{(i,j)} \sigma_i \sigma_j}}{Z} \tag{2.3.5}$$

where the sum is taken over nearest neighbours, the bold font indicates an N component vector which gives the value of σ on every site and Z is the normalising constant. To determine the normalising factor Z it is necessary to sum over all configurations. The first step is to pull out the common factor of e^{4KN}, which corresponds to the ground state of all spins being down (without loss of generality — the symmetric case of all spins being up is picked up as the multiplicative pre-factor of 2 in equation 2.3.6). Then systematically enumerate the corrections, starting from the configuration where one flip is up, which carries an energy penalty of $-8K$, reflecting the fact that each node interacts with 4 neighbours, and the sum total of the contribution from these 4 interactions will flip from $4K$ to $-4K$.

$$Z = 2\mathrm{e}^{4NK}\left[1 + N\mathrm{e}^{-8K} + \ldots\right] \tag{2.3.6}$$

Higher order terms can be defined in the same way. This means that the free energy per spin, proportional to $-\frac{1}{N}\ln Z$, is N independent for large N and can be estimated by taking a Taylor expansion of the logarithm, after having calculated the required number of terms for the desired accuracy.

The high-temperature expansion, which corresponds to small K, begins with a simple identity

$$\mathrm{e}^{K\sigma_i\sigma_j} = \frac{\mathrm{e}^{K} + \mathrm{e}^{-K}}{2} + \frac{\mathrm{e}^{K} + \mathrm{e}^{-K}}{2}\sigma_i\sigma_j = \cosh K(1 + t\sigma_i\sigma_j) \tag{2.3.7}$$

where $t = \tanh K$ will be the expansion parameter. This allows us to re-write

$$Z = \sum_{\{\sigma_i\}} \mathrm{e}^{K \sum_{(i,j)} \sigma_i \sigma_j} = (\cosh K)^N \sum_{\boldsymbol{\sigma}} \prod_{(i,j)} (1 + t\sigma_i \sigma_j) \tag{2.3.8}$$

Summing over all possible N-site vectors $\boldsymbol{\sigma}$ will cancel any odd powers appearing in the product. Non-cancelling terms can be identified as closed paths (e.g. the coefficient of t^4 will identify the number of ways in which 4 sites can be picked, so that sequentially moving from a vertex to one of its neighbours can define a closed path through these nodes). Calculating the coefficients becomes a problem of enumerating the number of closed paths that can be embedded in a lattice.

2.3.3 Replica method

In this section we will briefly present the rationale for the replica method — based on the keynote book by Mézard et al. [1987] — and then work through a simple illustrative example. So far, our examples have had fixed values for the coupling parameter J. Suppose that instead the values of the coupling parameter were drawn from some distribution $P[J]$, and we wanted to compute the average value of the free energy. The replica method is based on the identity $\sum_J P[J] \log Z = \langle \log Z \rangle = \lim_{n \to 0} \frac{1}{n} \log \langle Z^n \rangle$. This is justified by using the relation $A^n = 1 + n \ln A + O\left(n^2\right)$ for small n. It then follows that

$$\frac{1}{n} \ln(\langle Z^n \rangle) = \frac{1}{n} \ln(\langle 1 + n \ln Z \rangle) = \frac{1}{n} \ln(1 + n \langle \ln Z \rangle) = \langle \ln Z \rangle + O(n) \tag{2.3.9}$$

which gives the required relation. The method has been criticised for being non-rigorous, due to the somewhat unnatural approach of doing integer operations, and then doing an analytic continuation into the real numbers. However, subsequent authors have strengthened the theoretical foundations, and re-derived some of the replica results via more classical (albeit long and complicated) routes. There is now little doubt that this is a sound approach to obtain relatively simple solutions to otherwise prohibitively difficult and technical calculations.

2.3.3.1 Worked example using the replica approach (Yedidia [1992])

This worked example, taken from Yedidia [1992], considers a particle governed by Hamiltonian $H = \frac{r^2}{2} - fr$ where f is chosen from the probability distribution $p(f) = \frac{\mathrm{e}^{-f^2/2f_0^2}}{\sqrt{2\pi f_0^2}}$. The free energy $\bar{F}$ averaged over the entire ensemble will be computed using a direct route and the replica

route. This is quite a straightforward example, where the direct route works well. For more complicated examples, the direct route may be impossible — and hence the replica approach provides a complicated but precise approach to obtain a solution.

To directly calculate $\bar{F} \equiv \int_{-\infty}^{\infty} \mathrm{d}f p(f) F_f$ where $F_f = -\frac{1}{\beta} \ln Z_f$ and $Z_f = \int_{-\infty}^{\infty} \mathrm{e}^{-\beta H(r)} dr$ it is necessary to first complete the square on the Hamiltonian to achieve a Gaussian form. It then follows that $Z_f = \mathrm{e}^{f^2/2T} \sqrt{2\pi T}$ and $\bar{F} = -\frac{f_0^2}{2} - \frac{T}{2} \ln(2\pi T)$.

To approach this as a replica calculation, instead of doing the r integration first, it is necessary to start with the disorder average represented by the f integration. The replica identity can be used to write

$$\bar{F} = -T \int_{-\infty}^{\infty} \mathrm{d}f p(f) \ln Z_f \quad = -T \lim_{n\to 0} \tfrac{1}{n} \ln \left[\int_{-\infty}^{\infty} \mathrm{d}f p(f) Z_f^n \right] \tag{2.3.10}$$

Labelling the replicas with an index a which can run from 1 to n means that the problem is now

$$\bar{F} = -T \lim_{n\to 0} \frac{1}{n} \ln \left[\int_{-\infty}^{\infty} \frac{\mathrm{d}f}{\sqrt{2\pi f_0^2}} \mathrm{e}^{-f^2/2f_0^2} \int_{-\infty}^{\infty} \prod_{a=1}^{n} \mathrm{d}r_a \exp\left(-\frac{1}{T} \left(\sum_{a=1}^{n} \frac{r_a^2}{2} - f r_a \right) \right) \right] \tag{2.3.11}$$

Completing the square and performing Gaussian f integration reaches

$$\bar{F} = -T \lim_{n\to 0} \frac{1}{n} \ln \left[\int_{-\infty}^{\infty} \prod_{a=1}^{n} dr_a exp\left(-\frac{1}{T} \left(\sum_{a=1}^{n} \frac{r_a^2}{2} - \frac{f_0^2}{2T} \sum_{a=1}^{n} \sum_{b=1}^{n} r_a r_b \right) \right) \right]$$

By averaging over disorder the original problem has been converted into the mathematically equivalent problem of a system of n particles with no disorder, interacting according to the effective Hamiltonian

$$H_{eff} = \sum_{a=1}^{n} \frac{r_a^2}{2} - \frac{f_0^2}{2T} \sum_{a=1}^{n} \sum_{b=1}^{n} r_a r_b = \frac{1}{2} \sum_{a=1}^{n} \sum_{b=1}^{n} (G^{-1})_{ab} r_a r_b$$

where $(G^{-1})_{aa} = 1 - f_0^2/T$ and $(G^{-1})_{a\neq b} = -f_0^2/T$. This is a standard Gaussian integral.

$$\bar{F} = -T \lim_{n\to 0} \frac{1}{n} \ln \left(\sqrt{(2\pi T)^n \det G} \right) = -\frac{T}{2} \ln(2\pi T) - \frac{T}{2} \lim_{n\to 0} \frac{1}{n} Tr \ln G$$

The term $\ln G$ is evaluated by deriving the rules for multiplying matrices of this type, and then by a Taylor expansion. The final answer is, of course, the same: $\bar{F} = -\frac{f_0^2}{2} - \frac{T}{2} \ln(2\pi T)$.

2.3.4 Computer simulation

One approach to learn about average properties of complex systems is to simulate them via a computer. Monte Carlo is the term used for a strategy where a large number of parallel

realisations of a system are obtained via random sampling. Average properties of the ensemble can then be estimated as average properties of the (very many) end points of the simulation. The term generally appears juxtaposed with the phrase Markov Chain. A (discrete) Markov Chain is a random process moving in steps through time-points $t = 0, 1, 2 \ldots$, with the particular property that the probability of being in a particular state at time $t + 1$ depends only on the state at time t, and has no memory of the trajectory before that point. A Markov chain is described as ergodic if it is possible to move between any two states in the state-space in a finite number of steps. In chapter 6 we discuss in detail the properties of one particular Markov chain process on networks.

Markov Chain Monte Carlo (MCMC) is a very powerful technique. Its limitation is that many simulations need to be carried out to draw statistically valid conclusions. Hence analytical approaches are preferable where they are available, particularly when dealing with very large systems. However, since biological applications are often highly bespoke, computer simulations of null model networks plays an important role in building understanding. Problems that require effectively the enumeration of the ensemble, such as those covered in chapters 3 to 5, are essentially impossible to complete with computer simulations for large networks. In these cases, analytical techniques are essential.

2.4 Bridge from ferromagnets to biological networks

Below we briefly trace the development of the mathematical techniques which have taken the state of the art from modelling a one-dimensional ferromagnet, to modelling complex biological networks. To quote from Mézard et al. [1987]

> "Is it possible that what looks like goal oriented organisation with subtle relationships leading to amazing ways of reaching efficiency can in any way be seen as a maximally disordered system?"

To describe this in a different way, the underlying unstated orthodoxy is that the complex system (the network) has an underlying determined part (i.e.some rules or constraints), and the remainder is viewed as being completely random. The justification for this viewpoint given in Mézard et al. [1987] is that

> "The biosystem is much larger than has many more variables than the ones that have been put under control by the genetic code."

They also argue that this view is consistent with the fact that natural selection works through competition between random mutations. In this way they justify the analogy between biological systems and physical systems — which will typically also have macroscopic variables under control, and microscopic variables that are random. Whether or not it ultimately proves valid to view biological systems as having a strict divide between controlled and random elements, this is a philosophy which lends itself to rational design of predictive models. Our observations may in fact not be control/randomness — but rather randomness with correlations that we do not yet see — or controls which are sometimes less strict. However, the role of model design is to simplify as well as to capture — and a work program of isolating material control factors and then quantifying the amount of control they exert is thus a rational and practical way to improve our understanding.

Placing an Ising Model on a network (e.g. Dorogovtsev et al. [2002], Leone et al. [2002], Bianconi [2002]) provides an interesting intellectual challenge, and also works as a model of a process on a network. Bringing in more complicated networks, with controlled topologies indicated a shift in focus to study the network itself. The papers by Annibale et al. [2009], Bianconi [2009], Bianconi et al. [2008], Bianconi [2008] re-framed the question to using entropy for assessing the role of each constraint on a random graph ensemble. The authors of West et al. [2012] have published extensively, with a similar view to the problem as will be set out in this thesis. The main relevant papers are briefly described in this paragraph. Anand and Bianconi [2010] used the cavity approach to derive results for the entropy of simple networks under linear constraints. West et al. [2012]'s approach is to integrate gene expression data with a protein interaction network to generate a weighted protein interaction network. They define the network entropy of a node i to be $S_i = -\frac{1}{\log k_i} \sum_{(i,j)} p_{ij} \log p_{ij}$, where the sum is taken over neighbouring nodes, and the weights are taken from overlaying gene expression data onto the protein-protein interaction network. They find that cancer is characterised by an increase in this measure of entropy, and moreover that it is possible to distinguish metastasising and non-metastasising cancers through this approach. Bianconi [2013] defined a new topological structure 'multiplex', which is a topology characterised by the nodes taking part in several layers simultaneously, and formulate the concept of network entropy in this context. Anand and

Bianconi [2009] discusses the connections between the Shannon entropy, the Gibbs entropy and the von Neumann entropy (as promoted by Passerini and Severini [2008] and related papers). In Bianconi et al. [2009] and other papers by the same authors, the problem is posed in terms of nodes having associated hidden variables (e.g. community affiliations). Bianconi et al. [2009] specifically used an entropy based measure to quantify the dependence of a network's structure on a given set of features. Bianconi [2009] also looked at entropies of constrained random graph ensembles, although it did not reach a closed form solution for any but the simplest cases. One of the points discussed in this paper was that, in their observation, the power law degree distribution had lower entropy than the Poissonian one. Bianconi [2008] makes an approximation for the entropy of network ensembles with a given degree structure, degree correlation and community structure. Bianconi et al. [2008] were the first to approach the question of generalised degrees. The terminology used was 'hierarchically constrained topologies', and the problem was formulated for generalised degrees to an arbitrary depth. In chapter 4 we begin with the self-consistency relationship derived in this paper, and then set out two novel approaches to take the work further.

The immediate starting point for this thesis is the work by Annibale et al. [2009] which calculated precise leading order expressions for entropies and Kullback-Leibler distances for random graph ensembles defined in terms of degree distribution and degree correlations. The strategy can be summarised as using the tools of statistical mechanics to count the number of graphs in (large) constrained ensembles, and use this as a basis to make rational comment for the underlying network topologies. Instead of using a series expansion or an approximation, they defined the topology through a kernel which profited from the finite connectivity regime (see equation 3.1.4 for an example of this technique used within this thesis). This allowed the efficient elimination of non-leading order terms. From there, various manipulations and the application of the method of steepest descent led to the results which are summarised in table 3.1. A related paper (Fernandes et al. [2010]) used these techniques in order to demonstrate that networks derived by the same experimental method were topologically similar.

2.5 Review of relevant literature on applications of information theory

The use of statistical mechanics to quantify the information content of network structure is well established. Ferrer and Solé [2003] studied the 'degree entropy' — which they defined as $H = -\sum_k p(k) \log p(k)$ - in order to draw conclusions about network 'optimisation'. They charted a relationship between this quantity and how complex the network appeared to visual inspection. Demetrius and Manke [2005] use a similar definition of 'network entropy' and argue that network entropy is a quantitative measure of robustness. They go on to postulate that entropy is a selective criterion for Darwinian evolution. The same authors published Manke et al. [2006], investigating the link between network entropy and lethality (see chapter 9 for further exploration of this idea).

Passerini and Severini [2008] work with the normalised combinatorial Laplacian of a graph with eigenvalues λ_i. They define the von Neumann entropy of a graph as $S(G) = -\sum_{i=1}^{n} \lambda_i \log_2 \lambda_i$. They observe that this quantity may be interpreted as a measure of regularity, which tends to be larger in relation to the number of connected components, long paths and non-trivial symmetries. With their approach, regular graphs have maximal entropy, whereas large cliques minimize entropy compared to other networks with the same number of edges.

Wang et al. [2006] studied the robustness of scale-free networks to random failures via the entropy of the degree distribution. They argued that the entropy of the degree distribution is an effective measure of network's resilience to random failures. Lezon et al. [2006] constructed a gene interaction network based on the principle of entropy maximization to identify the gene interaction network with the highest probability of giving rise to experimentally observed transcription profiles.

Outside of a network context, Ritchie et al. [2008] studied the expression profiles of cancerous and non-cancerous tissues. They hypothesised that impairment of the transcriptional or post-transcriptional control machinery in cancer or other diseases should result in the loss of a tissue-specific expression patterns and that this loss can be measured by a gain of entropy in the expression pattern of isoforms of a given gene. It fits into a repeated idea where entropy type measures can be correlated with the ability of the system to perform a function. A similar approach could be taken forward in the networks context by, for example, comparing the entropy

of diseased and healthy cellular networks.

Wang et al. [2006], Ferrer and Solé [2003], Lezon et al. [2006] and other authors define degree entropy without particularly justifying why this is a meaningful calculation in this context. There is a tendency for the terminology to become somewhat loose — using 'degree entropy' and 'network entropy' synonymously. It is evident — and will be explored in great detail in this thesis — that a degree distribution is only a starting point to define the topology of a network. Hence the calculations presented by these authors are analytically clear and insightful, but in no way definitive.

CHAPTER 3

ENTROPY OF CONSTRAINED DIRECTED GRAPH ENSEMBLES

This thesis aims to generate new mathematical tools with which to quantify the macroscopic topological structure of large directed networks. In this chapter, this is achieved via a statistical mechanical analysis of constrained maximum entropy ensembles of directed random graphs with prescribed joint distributions for in- and out-degrees and prescribed degree-degree correlation functions. We calculate new exact and explicit formulae for the leading orders in the system size of the Shannon entropies and complexities of these ensembles, and for information-theoretic distances.

A limitation of Annibale et al. [2009] was that it only dealt with nondirected networks and graphs. Extending the methods in Annibale et al. [2009] to directed networks will enable their application to important new problems especially in cellular biology. Other applications could include the analysis and control of communication and computation networks. To understand the processes driving a cell it is necessary to go beyond studying individual genes; one needs to study their interactions. Information on how genes interact within the cell is commonly represented by a directed graph: the gene regulation network. High-throughput methods have generated a wealth of data on gene regulation. We now need powerful mathematical tools to

analyse these data. By focussing on which properties are the most important to the structure of the biological signalling network, we can envisage being able to postulate mechanisms for how the network evolved and came to fulfil its function, and build better models for such networks.

Evaluating the fit of a network model to network data is often seen as a formidable computational challenge (see e.g. Kuchaiev et al. [2010]), which is usually overcome by looking at fit based on comparing network properties. Our approach gives a rigorous quantitative method for prioritising network properties; this is important as different properties might promote different potential models. Degree-degree correlation refers to the joint distribution of connected nodes. Projections of degree-degree correlation include assortativity (the Pearson correlation coefficient of the degrees of connected nodes, as promoted by, for example, Newman [2002]), average nearest neighbour for a node with given degree k, or rich club connectivity (defined by Zhou and Mondragón [2004] as the ratio of links occurring within a set of nodes with degree greater than some given k, to the total possible number of links within such a set).

The specific quantities calculated in this chapter are: the Shannon entropy and complexity of directed graph ensembles with controlled degree distributions; the Shannon entropy and complexity of directed graph ensembles with controlled degree distributions and controlled degree-degree correlation functions; and, the symmetrised Kullback-Leibler distance between pairs of such ensembles. For each of these we calculate the leading orders in the network size, expressed in terms of the degree distributions and degree-degree correlation functions of the ensembles concerned. We illustrate the use of our results in Chapter 7 with applications to experimental data on gene regulation networks.

We adopt the following notation conventions. Each directed graph with N nodes is defined by a matrix $\boldsymbol{c} = \{c_{ij}\}$, with entries $c_{ij} \in \{0, 1\}$ indicating whether ($c_{ij} = 1$) or not ($c_{ij} = 0$) there is a directed arc from node j to node i. For each node i we define the in- and out-degrees as $k_i^{\text{out}}(\boldsymbol{c}) = \sum_j c_{ji}$ and $k_i^{\text{in}}(\boldsymbol{c}) = \sum_j c_{ij}$; in nondirected graphs such as in Annibale et al. [2009] one would have had $k_i^{\text{in}}(\boldsymbol{c}) = k_i^{\text{out}}(\boldsymbol{c})$ for all i. We write the pair of degrees at a site i as $\vec{k}_i(\boldsymbol{c}) = \left(k_i^{\text{in}}(\boldsymbol{c}), k_i^{\text{out}}(\boldsymbol{c})\right)$. Boldface letters will represent ordered sets with N elements, such as $\boldsymbol{k}^{\text{in}} = (k_1^{in}, \ldots, k_N^{in})$, or $\boldsymbol{k}^{\text{in}}(\boldsymbol{c}) = (k_1^{\text{in}}(\boldsymbol{c}), \ldots, k_N^{\text{in}}(\boldsymbol{c}))$.

3.1 Directed graphs with controlled in- and out-degree distributions

Here we calculate the Shannon entropy of an ensemble of directed random graphs constrained by a common joint distribution of in- and out-degrees. Via suitable adaptations of the methods developed for nondirected networks, we achieve a standard path-integral form to which we can apply the method of steepest descent. This leads to an elegant analytical expression for the entropy of the ensemble in the leading orders in N. The key term takes the form of a Kullback-Leibler distance between the imposed joint degree distribution and the Poissonnian one that would have been found upon generating directed arcs independently.

3.1.1 Definition of the problem

We consider an ensemble of directed random graphs, where degree pairs $\vec{k}_i = (k_i^{\rm in}, k_i^{\rm out})$ are for each node i drawn independently from a specified joint degree distribution $p(\vec{k})$

$$p(\boldsymbol{c}) = \sum_{\vec{k}_1\ldots\vec{k}_N} \Big[\prod_i p(\vec{k}_i)\Big] p(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N) \tag{3.1.1}$$

$$p(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N) = \frac{\prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})}}{Z(\vec{k}_1\ldots\vec{k}_N)}, \qquad Z(\vec{k}_1\ldots\vec{k}_N) = \sum_{\boldsymbol{c}} \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \tag{3.1.2}$$

For this ensemble we want to find the Shannon entropy per node $S = -N^{-1}\sum_{\boldsymbol{c}} p(\boldsymbol{c}) \log p(\boldsymbol{c})$, which informs us about the effective number $\mathcal{N} = \exp(NS)$ of graphs in the ensemble and the complexity of directed graphs with the imposed degree statistics $p(\vec{k})$. Upon substituting equation 3.1.2 into the entropy formula, and after some simple manipulations and use of the law of large numbers, one finds that the entropy per node takes the form

$$S = \frac{1}{N}\sum_{\vec{k}_1\ldots\vec{k}_N} \Big[\prod_i p(\vec{k}_i)\Big] \log Z(\vec{k}_1\ldots\vec{k}_N) - \sum_{\vec{k}} p(\vec{k}) \log p(\vec{k}) + \epsilon_N \tag{3.1.3}$$

where $\epsilon_N \to 0$ as $N \to \infty$. To make the first term in this expression more tractable, we transform $Z(\vec{k}_1\ldots\vec{k}_N)$ into an average involving an alternative measure. If we denote the average degree by $\bar{k} = N^{-1}\sum_i k_i^{\rm in} = N^{-1}\sum_i k_i^{\rm out}$, we may define the measure

$$\begin{aligned} w(\boldsymbol{c}|\bar{k}) &= \prod_{ij} \Big[\frac{\bar{k}}{N}\delta_{c_{ij},1} + \Big(1-\frac{\bar{k}}{N}\Big)\delta_{c_{ij},0}\Big] \\ &= \Big[1-\frac{\bar{k}}{N}\Big]^{N(N-1)} \Big[\frac{\bar{k}/N}{1-\bar{k}/N}\Big]^{N\bar{k}(\boldsymbol{c})} \equiv W(\bar{k}, \bar{k}(\boldsymbol{c})) \end{aligned} \tag{3.1.4}$$

This idea was developed in in Annibale et al. [2009], where it is argued why the insertion of this measure had no influence on the averages being calculated. Since this measure depends on the graph $\boldsymbol{c}$ via $\bar{k}(\boldsymbol{c})$ only, we can write the partition function $Z(\vec{k}_1 \dots \vec{k}_N)$ in terms of an average over the measure 3.1.4

$$Z(\vec{k}_1 \dots \vec{k}_N) = \frac{1}{W(\bar{k},\bar{k})} \sum_{\boldsymbol{c}} w(\boldsymbol{c}|\bar{k}) \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \tag{3.1.5}$$

Introducing the notation $\langle f(\boldsymbol{c})\rangle_{\bar{k}} = \sum_{\boldsymbol{c}} w(\boldsymbol{c}|\bar{k}) f(\boldsymbol{c})$ to represent averages over the measure 3.1.4 with average connectivity $\bar{k}$, the entropy per node can be written as

$$\begin{aligned} S &= \frac{1}{N} \sum_{\vec{k}_1 \dots \vec{k}_N} \Big[\prod_i p(\vec{k}_i) \Big] \log \Big\langle \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \Big\rangle_{\bar{k}} - \sum_{\vec{k}} p(\vec{k}) \log p(\vec{k}) \\ &\quad - \frac{1}{N} \sum_{\vec{k}_1 \dots \vec{k}_N} \Big[\prod_i p(\vec{k}_i) \Big] \log \left[\Big[1 - \frac{\bar{k}}{N} \Big]^{N(N-1)} \Big[\frac{\bar{k}/N}{1-\bar{k}/N} \Big]^{N\bar{k}} \right] + \epsilon_N \\ &= \frac{1}{N} \sum_{\vec{k}_1 \dots \vec{k}_N} \Big[\prod_i p(\vec{k}_i) \Big] \log \Big\langle \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \Big\rangle_{\bar{k}} - \sum_{\vec{k}} p(\vec{k}) \log p(\vec{k}) \\ &\qquad + \bar{k}[\log(N/\bar{k}) + 1] + \epsilon_N \end{aligned} \tag{3.1.6}$$

with $\lim_{N\to\infty} \epsilon_N = 0$, and with $\bar{k} = \sum_{\vec{k}} k^{\text{in}} p(\vec{k}) = \sum_{\vec{k}} k^{\text{out}} p(\vec{k})$. All the challenge of the problem is thus contained in the first term of equation 3.1.6

$$\phi = \frac{1}{N} \sum_{\vec{k}_1 \dots \vec{k}_N} \Big[\prod_i p(\vec{k}_i) \Big] \log \Big\langle \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \Big\rangle_{\bar{k}} \tag{3.1.7}$$

3.1.2 Entropy evaluation

Using Fourier representations of the Kronecker deltas in equation 3.1.7 and some straightforward manipulations brings us to

$$\phi = \frac{1}{N} \sum_{\vec{k}_1 \dots \vec{k}_N} \Big[\prod_i p(\vec{k}_i) \Big] \log \int_{-\pi}^{\pi} \prod_i \left[\frac{\mathrm{d}\omega_i \mathrm{d}\psi_i}{4\pi^2} \mathrm{e}^{\mathrm{i}[\omega_i k_i^{\text{in}} + \psi_i k_i^{\text{out}}]} \right] L(\boldsymbol{\omega},\boldsymbol{\psi}) \tag{3.1.8}$$

$$L(\boldsymbol{\omega},\boldsymbol{\psi}) = \exp\left[\bar{k} N \Big(\frac{1}{N} \sum_i \mathrm{e}^{-\mathrm{i}\omega_i} \Big) \Big(\frac{1}{N} \sum_j \mathrm{e}^{-\mathrm{i}\psi_j} \Big) - \bar{k} N + O(N^0) \right] \tag{3.1.9}$$

Introducing the quantities $R(\boldsymbol{\omega}) = N^{-1} \sum_i \mathrm{e}^{-\mathrm{i}\omega_i}$ and $S(\boldsymbol{\psi}) = N^{-1} \sum_i \mathrm{e}^{-\mathrm{i}\psi_i}$, and inserting $\int \mathrm{d}R \mathrm{d}S\ \delta[R - R(\boldsymbol{\omega})]\delta[S - S(\boldsymbol{\psi})]$ with δ-functions written in integral form, allows us to write

$$L(\boldsymbol{\omega},\boldsymbol{\psi}) = \int \frac{\mathrm{d}R \mathrm{d}\hat{R} \mathrm{d}S\, \mathrm{d}\hat{S}}{4\pi^2/N^2} \mathrm{e}^{N\left[\mathrm{i}(\hat{R}R + \hat{S}S) + \bar{k}(RS-1)\right] + O(N^0)} \times \prod_i \mathrm{e}^{-\mathrm{i}[\hat{R}\mathrm{e}^{-\mathrm{i}\omega_i} + \hat{S}\mathrm{e}^{-\mathrm{i}\psi_i}]} \tag{3.1.10}$$

Substituting this back into ϕ, using the law of large numbers, then gives

$$\phi \;=\; \frac{1}{N}\sum_{\vec{k}_1\ldots\vec{k}_N}\Big[\prod_i p(\vec{k}_i)\Big]\log\int \mathrm{d}R\mathrm{d}\hat{R}\mathrm{d}S\,\mathrm{d}\hat{S}\;\mathrm{e}^{N\Psi(R,\hat{R},S,\hat{S})+O(\log N)} \tag{3.1.11}$$

where

$$\Psi(R,\hat{R},S,\hat{S}) = \mathrm{i}(\hat{R}R+\hat{S}S)+\sum_{k^{\rm in}} p(k^{\rm in})\log\int_{-\pi}^{\pi}\frac{\mathrm{d}\omega}{2\pi}\mathrm{e}^{\mathrm{i}[\omega k^{\rm in}-\hat{R}\mathrm{e}^{-\mathrm{i}\omega}]}+\sum_{k^{\rm out}} p(k^{\rm out})\log\int_{-\pi}^{\pi}\frac{\mathrm{d}\psi}{2\pi}\mathrm{e}^{\mathrm{i}[\psi k^{\rm out}-\hat{S}\mathrm{e}^{-\mathrm{i}\psi}]}$$

The average in 3.1.11 over degree sequences is now obsolete since the argument depends in leading order in N on their distribution only, and 3.1.11 can be evaluated by steepest descent

$$\lim_{N\to\infty}\phi \;=\; \mathrm{extr}_{R,\hat{R},S,\hat{S}}\,\Psi(R,\hat{R},S,\hat{S}\,] \tag{3.1.12}$$

We can simplify Ψ by doing the remaining integrals, using

$$\int_{-\pi}^{\pi}\frac{\mathrm{d}\omega}{2\pi}\mathrm{e}^{\mathrm{i}[\omega k-A\mathrm{e}^{-\mathrm{i}\omega}]} = \sum_{m\geq 0}\frac{(-\mathrm{i}A)^m}{m!}\int_{-\pi}^{\pi}\frac{\mathrm{d}\omega}{2\pi}\mathrm{e}^{\mathrm{i}\omega(k-m)} = \frac{(-\mathrm{i}A)^k}{k!} \tag{3.1.13}$$

Hence

$$\begin{aligned}\Psi(R,\hat{R},S,\hat{S}) \;=\;& \mathrm{i}(\hat{R}R+\hat{S}S)+\bar{k}(RS-1)\\ &+\sum_{k^{\rm in}} p(k^{\rm in})\log[(-\mathrm{i}\hat{R})^{k^{\rm in}}/k^{\rm in}!]\\ &+\sum_{k^{\rm out}} p(k^{\rm out})\log[(-\mathrm{i}\hat{S})^{k^{\rm out}}/k^{\rm out}!]\end{aligned} \tag{3.1.14}$$

Differentiation of Ψ gives the following saddle-point equations

$$-\mathrm{i}\hat{R} = \bar{k}S, \qquad -\mathrm{i}\hat{S} = \bar{k}R \tag{3.1.15}$$

$$\mathrm{i}R\hat{R}+\bar{k}=0, \qquad \mathrm{i}S\hat{S}+\bar{k}=0 \tag{3.1.16}$$

We conclude that $RS = 1$, and hence at the saddle-point we have

$$\Psi(R,\hat{R},S,\hat{S}) \;=\; \sum_{k^{\rm in}} p(k^{\rm in})\log\pi_{\bar{k}}(k^{\rm in})+\sum_{k^{\rm out}} p(k^{\rm out})\log\pi_{\bar{k}}(k^{\rm out}) \tag{3.1.17}$$

with the Poissonnian degree distribution $\pi_{\bar{k}}(k) = \mathrm{e}^{-\bar{k}}\bar{k}^k/k!$.

3.1.3 Final analytical expression for the entropy of the ensemble

The intermediate result 3.1.17 can now be substituted back into the expression for the entropy of the constrained random graph ensemble defined in equation 3.1.6, giving

$$S \;=\; \bar{k}[\log(N/\bar{k})+1]-\sum_{k^{\rm in},k^{\rm out}} p(k^{\rm in},k^{\rm out})\log\left(\frac{p(k^{\rm in},k^{\rm out})}{\pi_{\bar{k}}(k^{\rm in})\pi_{\bar{k}}(k^{\rm out})}\right)+\zeta_N \tag{3.1.18}$$

where $\bar{k}$ is the average connectivity, N is the number of nodes in the network, $p(k^{\text{in}}, k^{\text{out}})$ is the degree distribution that constrained the random graph ensemble, and $\lim_{N\to\infty} \zeta_N = 0$.

The compact form of equation 3.1.18 enables us to interpret and understand this result for the entropy per node. For example, we can consider what the result would have been if the constraint on the ensemble had been less restrictive. If our ensemble was a maximum entropy ensemble on the space of all directed graphs, constrained by the average degree only (as opposed to the full joint in- and out-degree distribution), then the entropy per node would have been $S = \bar{k}[\log(N/\bar{k}) + 1]$. We see that this is identical to what we would obtain from 3.1.18 if the constraining degree-distribution was $p(k^{\text{in}}, k^{\text{out}}) = \pi(k^{\text{in}})\pi(k^{\text{out}})$; a trivial calculation confirms that in the maximum entropy ensemble with constrained average degree one indeed has $p(k^{\text{in}}, k^{\text{out}}) = \pi(k^{\text{in}})\pi(k^{\text{out}})$ for $N \to \infty$. Similarly, if we had chosen a maximum entropy ensemble of directed graphs constrained by a prescribed degree sequence (as opposed to a joint degree distribution), then the entropy would have taken the form

$$S = \bar{k}[\log(N/\bar{k}) + 1] + \sum_{k^{\text{in}},k^{\text{out}}} p(k^{\text{in}}, k^{\text{out}}) \log[\pi(k^{\text{in}})\pi(k^{\text{out}})] + \zeta_N \tag{3.1.19}$$

This value is seen to be simply equation 3.1.18 minus the Shannon entropy of the joint degree distribution $p(k^{\text{in}}, k^{\text{out}})$, reflecting the possible ways to relabel sites in the original ensemble; this freedom is removed once we specify the individual degrees rather than their distribution.

3.2 Directed graphs with controlled degree distributions and degree-degree correlation functions

We extend our calculation by imposing a degree-degree correlation function in addition to a degree distribution. Degree-degree correlations in networks are known to carry valuable information. They can give rise to properties such as *assortativity* or *disassortativity* and often reflect the algorithm responsible for a network's generation. One such algorithm, *preferential attachment*, is well illustrated by the World Wide Web, where pages are more likely to be linked to if they already have many pages linking to them. Preferential attachment models gained credibility by reproducing the typical fat tails often found in the degree distributions of real networks.

3.2.1 Definition of the problem

We now wish to generate graphs with degree pairs $(k_i^{\rm in}, k_i^{\rm out})$ again drawn independently from the distribution $p(\vec{k}) = p(k^{\rm in}, k^{\rm out})$, but now the link probabilities are modified by some function $Q(\vec{k}_i, \vec{k}_j|\bar{p})$ of the degrees of the nodes concerned, and their distribution, with $\vec{k}_i = (k_i^{\rm in}, k_i^{\rm out})$

$$p(\boldsymbol{c}|p,Q) = \sum_{\vec{k}_1\ldots\vec{k}_N}\Big[\prod_i p(\vec{k}_i)\Big]p(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q) \tag{3.2.1}$$

$$p(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q) = \frac{w(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q)\prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})}}{Z(\vec{k}_1\ldots\vec{k}_N,Q)} \tag{3.2.2}$$

$$Z(\vec{k}_1\ldots\vec{k}_N,Q) = \sum_{\boldsymbol{c}} w(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q)\prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \tag{3.2.3}$$

The difference with the graph ensemble in the previous section is the appearance of a new measure $w(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q)$, defined as

$$w(\boldsymbol{c}|\vec{k}_1\ldots\vec{k}_N,Q) = \prod_{i\neq j}\left[\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\delta_{c_{ij},1}+\Big(1-\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\Big)\delta_{c_{ij},0}\right] \tag{3.2.4}$$

with $Q(\vec{k}_i,\vec{k}_j|\bar{p}) \geq 0$ for all $(\vec{k}_i,\vec{k}_j)$, and with the distribution $\bar{p}(\vec{k}) = N^{-1}\sum_i \delta_{\vec{k},\vec{k}_i}$ and the average degree $\overline{k} = N^{-1}\sum_i k_i^{\rm in} = N^{-1}\sum_i k_i^{\rm out}$ of the imposed degree sequence. The objective of the measure 3.2.4 is to deform the graph probabilities so as to impose a specific correlation profile between the degrees of connected nodes, by a suitable choice of the kernel $Q(.,.)$. We take $Q(.,.)$ to be normalized such that $w(\boldsymbol{c}|\ldots)$ is asymptotically consistent with the average degree $\bar{k}$. This means that we demand $N^{-2}\sum_{ij}Q(\vec{k}_i,\vec{k}_j|\bar{p}) = 1$. Equivalently, $\sum_{\vec{k},\vec{k}'}\bar{p}(\vec{k})\bar{p}(\vec{k}')Q(\vec{k},\vec{k}'|\bar{p}) = 1$, which explains why $Q(.,.)$ depends on the distribution $\bar{p}$. The entropy per node S of our ensemble is

$$S = -\sum_{\boldsymbol{c}} p(\boldsymbol{c}|p,Q)\Omega(\boldsymbol{c}|p,Q) \tag{3.2.5}$$

$$\Omega(\boldsymbol{c}|p,Q) = N^{-1}\log p(\boldsymbol{c}|p,Q) \tag{3.2.6}$$

3.2.2 Entropy evaluation

In Appendix A we calculate the quantity 3.2.6 in leading orders in N, resulting in formula A.1.23. Substitution into expression 3.2.5 for the entropy, followed by doing the average over

$p(\boldsymbol{c}|p,Q)$ and some simple re-arranging of terms, then gives us

$$\begin{aligned} S &= \bar{k}[\log(N/\bar{k})+1] - \sum_{\vec{k}} p(\vec{k}) \log\Big[\frac{p(\vec{k})}{\pi_{\bar{k}}(k^{\rm in})\pi_{\bar{k}}(k^{\rm out})}\Big] \\ &\quad -\bar{k}\sum_{\vec{k},\vec{k}'} W(\vec{k},\vec{k}') \log\Big[\frac{R(\vec{k}|p,Q)Q(\vec{k},\vec{k}'|p)S(\vec{k}'|p,Q)}{W_1(\vec{k})W_2(\vec{k}')}\Big] + \tilde{\zeta}_N \end{aligned} \tag{3.2.7}$$

with $\lim_{N\to\infty}\tilde{\zeta}_N = 0$, $\pi_{\bar{k}}(k) = {\rm e}^{-\bar{k}}\bar{k}^k/k!$, and $\bar{k} = \sum_{\vec{k}} p(\vec{k})k^{\rm in} = \sum_{\vec{k}} p(\vec{k})k^{\rm out}$. The kernel $W(\vec{k},\vec{k}')$ and its two marginals $W_{1,2}(\vec{k})$ in this expression are as defined in A.1.8,A.1.9,A.1.10, but now calculated for graphs from our ensemble 3.2.1. Similarly, the quantities $R(\vec{k}|p,Q)$ and $Q(\vec{k}|p,Q)$ are now solved from

$$R(\vec{k}) = \frac{p(\vec{k})k^{\rm in}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k},\vec{k}'|p)S(\vec{k}')} \quad \text{and} \quad S(\vec{k}) = \frac{p(\vec{k})k^{\rm out}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k}',\vec{k}|p)R(\vec{k}')} \tag{3.2.8}$$

in which the distribution $p(\vec{k})$, its associated average $\bar{k}$, as well as the kernel $Q(\vec{k},\vec{k}'|p)$, correspond to ensemble 3.2.1. Thus the correct normalization of the kernel $Q(.,.)$ is

$$\sum_{\vec{k},\vec{k}'} p(\vec{k})p(\vec{k})Q(\vec{k},\vec{k}'|p) = 1 \tag{3.2.9}$$

What remains is to express the distribution $W(\vec{k},\vec{k}'|p,Q)$ for ensemble 3.2.1 in terms of $\{p,Q\}$. This is done in Appendix A.2, resulting in equation A.2.3

$$\lim_{N\to\infty} W(\vec{k},\vec{k}') = R(\vec{k}|p,Q)Q(\vec{k},\vec{k}'|p)S(\vec{k}'|p,Q) \tag{3.2.10}$$

in which $R(\vec{k}|p,Q)$ and $S(\vec{k}|p,Q)$ are once more the solutions of 3.2.8, but now with $\tilde{p}(\vec{k})$ replaced by $p(\vec{k})$. Combination with equation 3.2.7 then gives us

$$\begin{aligned} S &= \bar{k}[\log(N/\bar{k})+1] - \sum_{\vec{k}} p(\vec{k}) \log\Big[\frac{p(\vec{k})}{\pi_{\bar{k}}(k^{\rm in})\pi_{\bar{k}}(k^{\rm out})}\Big] \\ &\quad -\bar{k}\sum_{\vec{k},\vec{k}'} W(\vec{k},\vec{k}') \log\Big[\frac{W(\vec{k},\vec{k}')}{W_1(\vec{k})W_2(\vec{k}')}\Big] + \tilde{\epsilon}_N \end{aligned} \tag{3.2.11}$$

with $\lim_{N\to\infty}\tilde{\epsilon}_N = 0$. Compared to the entropy per node 3.1.19 of ensembles where only the in-out degree distributions are imposed, we see that imposing in addition our new constraint, the specific degree-degree correlations as embodied by $W(\vec{k},\vec{k}')$, leads to a reduction of the entropy by an amount proportional to the mutual information of in-out degrees of connected nodes. An analogous result was derived in Annibale et al. [2009] for nondirected graphs. It can immediately be seen that if the in-out degrees of connected nodes are statistically independent, then the final non-vanishing term of equation 3.2.11 will be zero. Hence the entropy of the ensemble will in that case be as though the only constraint was the degree distribution.

3.3 Quantifying structural distance between networks

In this section we define and calculate an information theoretic distance between two directed networks A and B, with in-out degree distributions $p_A(\vec{k})$ and $p_B(\vec{k})$ and with degree-degree correlation functions $W_A(\vec{k},\vec{k}')$ and $W_B(\vec{k},\vec{k}')$. We generalize to the present context of directed graphs the choice made in Annibale et al. [2009] of the Jeffreys divergence (i.e. symmetrized Kullback-Leibler distance) per node of the two associated ensembles from our family 3.2.1

$$D_{AB} = \tfrac{1}{2N}\sum_{\boldsymbol{c}}\Big\{p(\boldsymbol{c}|p_A,Q_A)\log\Big[\tfrac{p(\boldsymbol{c}|p_A,Q_A)}{p(\boldsymbol{c}|p_B,Q_B)}\Big]+p(\boldsymbol{c}|p_B,Q_B)\log\Big[\tfrac{p(\boldsymbol{c}|p_B,Q_B)}{p(\boldsymbol{c}|p_A,Q_A)}\Big]\Big\} \tag{3.3.1}$$

D_{AB} is non-negative and equals zero only when both networks A and B belong to the same tailored graph ensemble (i.e. have equivalent constraints). Upon writing the Shannon entropies per node of the ensembles A and B as S_A and S_B, we have

$$D_{AB} = \frac{1}{2}(S_{AB}+S_{BA}-S_{AA}-S_{BB}) \tag{3.3.2}$$

where, using the abbreviation 3.2.6,

$$\begin{aligned} S_{AB} &= -\frac{1}{N}\sum_{\boldsymbol{c}}p(\boldsymbol{c}|p_A,Q_A)\log p(\boldsymbol{c}|p_B,Q_B) \\ &= -\sum_{\boldsymbol{c}}p(\boldsymbol{c}|p_A,Q_A)\Omega(\boldsymbol{c}|p_B,Q_B) \end{aligned} \tag{3.3.3}$$

with $\Omega(\boldsymbol{c}|p,Q)$ as defined in equation 3.2.6. We may now use result A.1.23 of Appendix A, but in doing so it is vital to keep track carefully of the labels (A,B) of the degree distributions and kernels. In particular, according to 3.3.3 we must make in equation A.1.23 the substitutions $p(\vec{k}|\boldsymbol{c})\to p_A(\vec{k})$, $W(\vec{k},\vec{k}'|\boldsymbol{c})\to W_A(\vec{k},\vec{k}')$, $p(\vec{k})\to p_B(\vec{k})$, and $Q(\vec{k},\vec{k}'|\tilde{p})\to Q_B(\vec{k},\vec{k}'|p_A)$. This leads us to

$$\begin{aligned} \lim_{N\to\infty}S_{AB} &= -\sum_{\vec{k}}p_A(\vec{k})\log p_B(\vec{k})-\bar{k}_A\Big[1+\log\big(\frac{\bar{k}_A}{N}\big)\Big]-\sum_{\vec{k}}p_A(\vec{k})\log(k^{\rm in}!k^{\rm out}!) \\ &\quad+\sum_{\vec{k}}p_A(\vec{k})k^{\rm in}\log\Big[\frac{p_A(\vec{k})k^{\rm in}}{R(\vec{k}|p_A,Q_B)}\Big]+\sum_{\vec{k}}p_A(\vec{k})k^{\rm out}\log\Big[\frac{p_A(\vec{k})k^{\rm out}}{S(\vec{k}|p_A,Q_B)}\Big] \\ &\quad-\bar{k}_A\sum_{\vec{k},\vec{k}'}W_A(\vec{k},\vec{k}')\log Q_B(\vec{k},\vec{k}'|p_A) \end{aligned} \tag{3.3.4}$$

in which $R(\vec{k}|p_A,Q_B)$ and $S(\vec{k}|p_A,Q_B)$ are to be solved from

$$R(\vec{k})=\frac{p_A(\vec{k})k^{\rm in}}{\bar{k}_A\sum_{\vec{k}'}Q_B(\vec{k},\vec{k}'|p_A)S(\vec{k}')}\quad\text{and}\quad S(\vec{k})=\frac{p_A(\vec{k})k^{\rm out}}{\bar{k}_A\sum_{\vec{k}'}Q_B(\vec{k}',\vec{k}|p_A)R(\vec{k}')} \tag{3.3.5}$$

Hence, upon assembling and combining the various terms in equation 3.3.2 and upon using relations such as A.1.9 A.1.10 and A.2.3 to simplify the result, we find

$$\begin{aligned} D_{AB} &= \frac{1}{2}\sum_{\vec{k}} p_A(\vec{k}) \log\Big[\frac{p_A(\vec{k})}{p_B(\vec{k})}\Big] + \frac{1}{2}\sum_{\vec{k}} p_B(\vec{k}) \log\Big[\frac{p_B(\vec{k})}{p_A(\vec{k})}\Big] \\ &+\frac{1}{2}\bar{k}_A \sum_{\vec{k},\vec{k}'} W_A(\vec{k},\vec{k}') \log\left[\frac{W_A(\vec{k},\vec{k}')}{R(\vec{k}|p_A,Q_B)Q_B(\vec{k},\vec{k}'|p_A)S(\vec{k}'|p_A,Q_B)}\right] \\ &+\frac{1}{2}\bar{k}_B \sum_{\vec{k},\vec{k}'} W_B(\vec{k},\vec{k}') \log\left[\frac{W_B(\vec{k},\vec{k}')}{R(\vec{k}|p_B,Q_A)Q_A(\vec{k},\vec{k}'|p_B)S(\vec{k}'|p_B,Q_A)}\right] \end{aligned} \tag{3.3.6}$$

According to A.2.3, the product $W_{AB}(\vec{k},\vec{k}') = R(\vec{k}|p_A,Q_B)Q_B(\vec{k},\vec{k}'|p_A)S(\vec{k}'|p_A,Q_B)$ equals the joint distribution of in- and out- degrees of connected nodes in an ensemble of the family defined in expression 3.2.1 that would have been obtained upon choosing the hybrid combination $\{p_A, Q_B\}$ of degree distribution and wiring kernel, where Q_B is normalized according to $\sum_{\vec{k},\vec{k}'} p_A(\vec{k})p_A(\vec{k}')Q_B(\vec{k},\vec{k}'|p_A) = 1$. Similarly, the product

$$W_{BA}(\vec{k},\vec{k}') = R(\vec{k}|p_B,Q_A)Q_A(\vec{k},\vec{k}'|p_B)S(\vec{k}'|p_B,Q_A) \tag{3.3.7}$$

would have been obtained for the ensemble $\{p_B, Q_A\}$. Thus we may write

$$\begin{aligned} \lim_{N\to\infty} D_{AB} &= \frac{1}{2}\sum_{\vec{k}} p_A(\vec{k}) \log\Big[\frac{p_A(\vec{k})}{p_B(\vec{k})}\Big] + \frac{1}{2}\sum_{\vec{k}} p_B(\vec{k}) \log\Big[\frac{p_B(\vec{k})}{p_A(\vec{k})}\Big] \\ &+\frac{1}{2}\bar{k}_A \sum_{\vec{k},\vec{k}'} W_A(\vec{k},\vec{k}') \log\Big[\frac{W_A(\vec{k},\vec{k}')}{W_{AB}(\vec{k},\vec{k}')}\Big] \\ &+\frac{1}{2}\bar{k}_B \sum_{\vec{k},\vec{k}'} W_B(\vec{k},\vec{k}') \log\Big[\frac{W_B(\vec{k},\vec{k}')}{W_{BA}(\vec{k},\vec{k}')}\Big] \end{aligned} \tag{3.3.8}$$

This appealing formula shows that $D_{AB} \geq 0$ for all choices of (A, B), with equality if and only if $W_A = W_B$; in the later case one automatically will have $W_{AB} = W_{BA} = W_A = W_B$. In the case where degree-degree correlations are absent from both networks one will find

$$W_{AB}(\vec{k},\vec{k}') = W_A(\vec{k},\vec{k}') = W_{1A}(\vec{k})W_{2A}(\vec{k}') \tag{3.3.9}$$

and formula 3.3.8 reduces to the Jeffreys divergence between the degree distributions p_A and p_B. Please refer to Roberts et al. [2011] for steps to resolve the meaning of the object $W_{BA}(\vec{k})$ and to incorporate it into a more practical form.

3.4 Tests and comparisons of the derived expressions

3.4.1 Simple special cases

If the in-degrees are statistically independent of the out-degrees, i.e. $p(\vec{k}) = p(k^{in})p(k^{out})$, the entropy per node 3.1.18 of the ensemble 3.1.1 with prescribed degree statistics but no degree correlations simplifies to

$$S = \bar{k}[\log\left(\frac{N}{\bar{k}}\right)+1] - \sum_{k^{\rm in}} p(k^{\rm in}) \log\left[\frac{p(k^{\rm in})}{\pi_{\bar{k}}(k^{\rm in})}\right] - \sum_{k^{\rm out}} p(k^{\rm out}) \log\left[\frac{p(k^{\rm out})}{\pi_{\bar{k}}(k^{\rm out})}\right] + \zeta_N \quad (3.4.1)$$

with $\lim_{N\to\infty} \zeta_N = 0$. This, according to Annibale et al. [2009], is the sum of the individual entropies of the 'out-graph' ensemble and the 'in-graph' ensemble, calculated as though they were considered as two separate nondirected networks. In ensembles with degree correlations, i.e. as defined in expression 3.2.1, with entropy per node as in 3.2.11, the additional term that represents the entropy reduction imposed by the degree correlations does not simplify as a result of assuming $p(\vec{k}) = p(k^{in})p(k^{out})$; the degree correlations can generate statistical relations between in- and out-degrees that are not visible in $p(\vec{k})$.

A regular directed graph is one where each node has the same in- and the same out-degree. Since for a well-defined directed graph, we also have $\sum_{\vec{k}} p(\vec{k})k^{\rm in} = \sum_{\vec{k}} p(\vec{k})k^{\rm out} = \overline{k}$, any regular directed graph must have $p(\vec{k}) = \delta_{\vec{k},(\bar{k},\bar{k})}$. This, in turn, implies also that $W(\vec{k},\vec{k}') = \delta_{\vec{k},(\bar{k},\bar{k})}\delta_{\vec{k}',(\bar{k},\bar{k})}$. So it is impossible to have degree correlations, and both equation 3.1.18 and 3.2.11 reduce to

$$S = \bar{k}[\log(N\bar{k}) - 1] - 2\log(\bar{k}!) + \zeta_N \quad (3.4.2)$$

3.4.2 Comparison of formulae for nondirected versus directed networks

It is instructive to give an overview of the similarities and differences between directed and nondirected graphs. Instead of entropies per node, we will also compare entropic results in terms of complexities. The degree complexity per node $C_{\rm deg}$ of a graph $\boldsymbol{c}$ is the difference between the entropy per node of the associated ensemble 3.1.1 and the value $S_0[\bar{k}]$ that is found for the entropy per node if only the average connectivity $\bar{k}$ is prescribed (i.e. for an ensemble with Poisson distributed degrees). The wiring complexity $C_{\rm wir}$ is the further entropy reduction that results if we go from the ensemble 3.1.1 to the ensemble 3.2.1 where also the degree-degree correlations are imposed. Our results can then be summarized as in table 3.1.

	Directed graphs	Nondirected graphs
$S_0[\bar{k}]$	$\bar{k}[\log(N/\bar{k})+1]$	$\frac{1}{2}\bar{k}[\log(N/\bar{k})+1]$
$C_{\rm deg}[p]$	$\sum_{\vec{k}} p(\vec{k}) \log\left[\frac{p(\vec{k})}{\pi_{\bar{k}}(k^{\rm in})\pi_{\bar{k}}(k^{\rm out})}\right]$	$\sum_k p(k)\log\left[\frac{p(k)}{\pi_{\bar{k}}(k)}\right]$
$C_{\rm wir}[p,W]$	$\bar{k}\sum_{\vec{k},\vec{k}'} W(\vec{k},\vec{k}')\log\left[\frac{W(\vec{k},\vec{k}')}{W_1(\vec{k})W_2(\vec{k}')}\right]$	$\frac{1}{2}\bar{k}\sum_{k,k'} W(k,k')\log\left[\frac{W(k,k')}{W(k)W(k')}\right]$

Table 3.1: Comparison of entropies and complexities of directed versus nondirected graphs. The entropy per node is given by $S[p,W] = S_0[\bar{k}] - C_{\rm deg}[p] - C_{\rm wir}[p,W]$, modulo finite size corrections. For ensembles in which only the average connectivity $\bar{k}$ is prescribed one would find the value $S_0[\bar{k}]$. The quantities $C_{\rm deg}[p]$ and $C_{\rm wir}[p,W]$ measure the entropy reductions caused by subsequently imposing a degree distribution p, and the joint distribution W of connected nodes, and can therefore be identified with the degree complexity and the wiring complexity of the typical graphs in our ensembles. In directed graphs $\vec{k} = (k^{\rm in}, k^{\rm out})$, where $k_i^{\rm in}(\boldsymbol{c}) = \sum_j c_{ij}$ and $k_i^{\rm out}(\boldsymbol{c}) = \sum_j c_{ji}$, and $W(\vec{k},\vec{k}') = (N\bar{k})^{-1}\sum_{ij} c_{ij}\delta_{\vec{k},\vec{k}_i}\delta_{\vec{k}',\vec{k}_j}$. In nondirected graphs one has only $k_i(\boldsymbol{c}) = \sum_j c_{ij}$, and $W(k,k') = (N\bar{k})^{-1}\sum_{ij} c_{ij}\delta_{k,k_i}\delta_{k',k_j}$. Please note that depending on context S is normalised 'per node' (chapters 3 and 4) or expressed directly (chapter 5).

Similarly we can compare the formulae for the information-theoretic distance D_{AB} between two networks $\boldsymbol{c}_A$ and $\boldsymbol{c}_B$, for directed versus nondirected ones. This gives in both cases $\lim_{N\to\infty} D_{AB} = D_{AB}^{\rm deg} + D_{AB}^{\rm wir} + D_{AB}^{\rm int}$, where $D_{AB}^{\rm deg}$ is the direct contribution from degree distribution dissimilarity, $D_{AB}^{\rm wir}$ is the direct contribution from degree-correlation dissimilarity, and $D_{AB}^{\rm int}$ accounts for the interference between degree statistics and the possible degree correlations that could be achieved. Our distance results can then be summarized in table 3.2.

The functions $\rho_{AB}(\vec{k})$ and $\sigma_{AB}(\vec{k})$ are solved from

$$\rho_{AB}(\vec{k}) = \sum_{\vec{k}'} \Pi_B(\vec{k},\vec{k}') W_{2A}(\vec{k}')\sigma_{AB}^{-1}(\vec{k}') \tag{3.4.3}$$

$$\sigma_{AB}(\vec{k}) = \sum_{\vec{k}'} \Pi_B(\vec{k}',\vec{k}) W_{1A}(\vec{k}')\rho_{AB}^{-1}(\vec{k}') \tag{3.4.4}$$

based on derivations in Roberts et al. [2011].

Repeating the calculation for nondirected graphs shows that only one function $\rho_{AB}(k)$ is required

	Directed graphs	Nondirected graphs
$D_{AB}^{\rm deg}$	$\frac{1}{2}\sum_{\vec{k}} p_A(\vec{k}) \log\left[\frac{p_A(\vec{k})}{p_B(\vec{k})}\right]$	$\frac{1}{2}\sum_{k} p_A(k) \log\left[\frac{p_A(k)}{p_B(k)}\right]$
	$+\frac{1}{2}\sum_{\vec{k}} p_B(\vec{k}) \log\left[\frac{p_B(\vec{k})}{p_A(\vec{k})}\right]$	$+\frac{1}{2}\sum_{k} p_B(k) \log\left[\frac{p_B(k)}{p_A(k)}\right]$
$D_{AB}^{\rm wir}$	$\frac{1}{2}\bar{k}_A \sum_{\vec{k},\vec{k}'} W_A(\vec{k},\vec{k}') \log\left[\frac{\Pi_A(\vec{k},\vec{k}')}{\Pi_B(\vec{k},\vec{k}')}\right]$	$\frac{1}{4}\bar{k}_A \sum_{k,k'} W_A(k,k') \log\left[\frac{\Pi_A(k,k')}{\Pi_B(k,k')}\right]$
	$+\frac{1}{2}\bar{k}_B \sum_{\vec{k},\vec{k}'} W_B(\vec{k},\vec{k}') \log\left[\frac{\Pi_B(\vec{k},\vec{k}')}{\Pi_A(\vec{k},\vec{k}')}\right]$	$+\frac{1}{4}\bar{k}_B \sum_{k,k'} W_B(k,k') \log\left[\frac{\Pi_B(k,k')}{\Pi_A(k,k')}\right]$
$D_{AB}^{\rm int}$	$\frac{1}{2}\bar{k}_A \sum_{\vec{k},\vec{k}'} W_A(\vec{k},\vec{k}') \log[\rho_{AB}(\vec{k})\sigma_{AB}(\vec{k}')]$	$\frac{1}{2}\bar{k}_A \sum_{k} W_A(k) \log \rho_{AB}(k)$
	$+\frac{1}{2}\bar{k}_B \sum_{\vec{k},\vec{k}'} W_B(\vec{k},\vec{k}') \log[\rho_{BA}(\vec{k})\sigma_{BA}(\vec{k}')]$	$+\frac{1}{2}\bar{k}_B \sum_{k} W_B(k) \log \rho_{BA}(k)$

Table 3.2: Comparison of the contributions to the distance $\lim_{N\to\infty} D_{AB} = D_{AB}^{\rm deg} + D_{AB}^{\rm wir} + D_{AB}^{\rm int}$, between graphs $\boldsymbol{c}_A$ and $\boldsymbol{c}_B$. Notation conventions are mostly as in the caption of table 3.1. The degree correlation ratios Π are defined as $\Pi(\vec{k},\vec{k}') = W(\vec{k},\vec{k}')/W_1(\vec{k})W_2(\vec{k}')$ (for directed graphs) and $\Pi(k,k') = W(\vec{k},\vec{k}')/W(k)W(k')$ (for nondirected graphs). The functions $\rho_{AB}(\vec{k})$ and $\sigma_{AB}(\vec{k})$ (for directed graphs) are the solutions of equations 3.4.3,3.4.4. The functions $\rho_{AB}(k)$ (for nondirected graphs) are to be solved from equation 3.4.5.

(or equivalently, $\rho_{AB} = \sigma_{AB}$), which is the solution of

$$\rho_{AB}(k) \quad = \quad \sum_{k'} \Pi_B(k,k') W_A(k') \rho_{AB}^{-1}(k') \tag{3.4.5}$$

3.5 Summary

In this chapter we have derived several mathematical results for directed random graph ensembles tailored to match chosen properties of real-world networks. We have calculated the Shannon entropy of ensembles constrained by a prescribed degree distribution, and of ensembles constrained by a prescribed degree-degree correlation function (which contains more detailed topological information than the degree distribution). Examples of calculations with parameters based on gene regulation networks is provided in chapter 7. We have also defined a rational information-theoretic distance measurement for comparing networks based on their degree distribution and degree-degree correlation.

CHAPTER 4

FURTHER GENERALISATIONS OF THE DEGREE CONSTRAINT

In this chapter we extend the work in the previous chapter by calculating in leading order, the Shannon entropies of three as yet unsolved families of random graph ensembles, constrained by: a bipartite constraint with imposed degree distributions in the two nodes sets, a neighbourhood distribution (where the neighbourhood of a node is defined as its own degree, plus the degree values of the nodes connected to it), and an imposed generalised degree distribution. The first two cases can be resolved exactly. The third case was already partially studied in Coolen et al. [2009], with only limited success, and here we require a plausible but as yet unproven conjecture to find an explicit formula for the entropy.

4.1 Definitions and notation

We consider ensembles of directed and nondirected random graphs. Each graph is defined by its adjacency matrix $\boldsymbol{c} = \{c_{ij}\}$, with $i, j \in \{1, \ldots, N\}$ and with $c_{ij} \in \{0, 1\}$ for all (i, j). Two nodes i and j are connected by a directed link $j \to i$ if and only if $c_{ij} = 1$. We put $c_{ii} = 0$ for all i. In nondirected graphs one has $c_{ij} = c_{ji}$ for all (i, j), so $\boldsymbol{c}$ is symmetric. The degree of a node

i in a nondirected graph is the number of its neighbours, $k_i = \sum_j c_{ij}$. In directed graphs we distinguish between in- and out-degrees, $k_i^{\rm in} = \sum_j c_{ij}$ and $k_i^{\rm out} = \sum_j c_{ji}$. They count the number of in- and out-bound links at a node i. A bipartite graph is one where the nodes can be divided into two disjoint sets, such that $c_{ij} = 0$ for all i and j that belong to the same set.

We define the set of neighbours of a node i in a nondirected graph as $\partial_i = \{j|\ c_{ij} = 1\}$. Hence $k_i = |\partial_i|$. To characterise a graph's topology near i in more detail we can define the generalised degree of i as the pair (k_i, m_i), where $m_i = \sum_j c_{ij} k_j$ counts the number of length-two paths starting in i. The concept of a generalised degree is discussed in Newman [2009]. Even more information is contained in the *local neighbourhood*

$$n_i \quad = \quad (k_i;\ \{\xi_i^s\}) \tag{4.1.1}$$

in which the ordered integers $\{\xi_i^s\}$ give the degrees of the k_i neighbours $j \in \partial_i$. See also Fig. 4.1. Since $m_i = \sum_{s \leq k_i} \xi_i^s$, the neighbourhood n_i provides more granular information that complements that in the generalised degree (k_i, m_i). We will use bold symbols when local topological parameters are defined for every node in a network, e.g. $\boldsymbol{k} = (k_1, \ldots, k_N)$ and $\boldsymbol{n} = ((k_1;\ \{\xi_1^s\}), \ldots, (k_N;\ \{\xi_N^s\}))$. Generalisation to directed graphs is straightforward. Here $\partial_i = \{j|\ c_{ij} + c_{ji} > 0\}$, and the local neighbourhood would be defined as $n_i = (\vec{k}_i;\ \{\vec{\xi}_i^s\})$ with the k_i pairs $\vec{\xi}_i^s = (k^{s,\rm in}, k^{s,\rm out})$ now giving both the in- and out-degrees of the neighbours of i.

Our tailored random graph ensembles will be of the following form, involving N built-in local (site specific) topological constraints of the type discussed above, which we will for now write generically as $X_i(\boldsymbol{c})$, and with the usual abbreviation $\delta_{\boldsymbol{a},\boldsymbol{b}} = \prod_i \delta_{a_i,b_i}$

$$p(\boldsymbol{c}) \quad = \quad \sum_{\boldsymbol{X}} p(\boldsymbol{X})\, p(\boldsymbol{c}|\boldsymbol{X}) \qquad p(\boldsymbol{X}) = \prod_i p(X_i) \tag{4.1.2}$$

$$p(\boldsymbol{c}|\boldsymbol{X}) \quad = \quad Z^{-1}(\boldsymbol{X}) \delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})}, \qquad Z(\boldsymbol{X}) = \sum_{\boldsymbol{c}} \delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})} \tag{4.1.3}$$

The values X_i for the local features are for each i drawn randomly and independently from $p(X)$, after which one generates a graph $\boldsymbol{c}$ randomly and with uniform probabilities from the set of graphs that satisfy the N demands $X_i(\boldsymbol{c}) = X_i$. The empirical distribution $p(X|\boldsymbol{c}) = N^{-1} \sum_i \delta_{X,X_i(\boldsymbol{c})}$ of local features will be random, but the law of large numbers ensures that for $N \to \infty$ it will converge to the chosen $p(X)$ for any graph realisation, and the above definitions guarantee that its ensemble average will be identical to $p(X)$ for any N,

$$\sum_{\boldsymbol{c}} p(\boldsymbol{c}) p(X|\boldsymbol{c}) \quad = \quad \frac{1}{N} \sum_i \sum_{\boldsymbol{X}} p(\boldsymbol{X}) \sum_{\boldsymbol{c}} \frac{\delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})}}{Z(\boldsymbol{X})} \delta_{X,X_i(\boldsymbol{c})} = p(X) \tag{4.1.4}$$

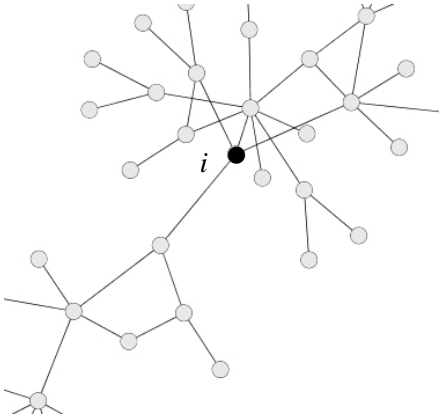


Figure 4.1: Illustration of our definitions of local topological characteristics in nondirected graphs. At the minimal level one specifies for each node i (black vertex in the picture) only the degree $k_i = |\partial_i| = \sum_j c_{ij}$ (the number of its neighbours). At the next level of detail one provides for each node the generalised degree (k_i, m_i), in which $m_i = \sum_{j\in\partial_i} k_j = \sum_j c_{ij}k_j$ is the number of length-two paths starting in i. This is then generalised to include the actual degrees in the set ∂_i, by giving $n_i = (k_i;\ \{\xi_i^s\})$ (the 'local neighbourhood'), in which the k_i integers $\{\xi_i^s\}$ give the degrees of the nodes connected to i. To avoid ambiguities we adopt the ranking convention $\xi_i^1 \leq \xi_i^2 \leq \ldots \leq \xi_i^{k_i}$. Note that $m_i = \sum_{j\in\partial_i} k_j = \sum_{s=1}^{k_i} \xi_i^s$.

If we aim to impose upon our graphs only a degree distribution we choose $X_i(\boldsymbol{c}) = k_i(\boldsymbol{c})$. Building in a distribution of generalised degrees corresponds to $X_i(\boldsymbol{c}) = (k_i(\boldsymbol{c}), m_i(\boldsymbol{c}))$. If we seek to prescribe the distribution of all local neighbourhoods (given by 4.1.1) we choose $X_i(\boldsymbol{c}) = n_i(\boldsymbol{c})$.

A further quantity which will play a role in subsequent calculations is the joint degree distribution of connected nodes. For nondirected graphs it is defined as

$$W(k, k'|\boldsymbol{c}) \quad = \quad \frac{\sum_{ij} c_{ij}\delta_{k,k_i}\delta_{k',k_j}}{\sum_{ij} c_{ij}} \tag{4.1.5}$$

and its average over the ensemble 4.1.2 is given by

$$W(k, k') \quad = \quad \sum_{\boldsymbol{X}} p(\boldsymbol{X}) \sum_{\boldsymbol{c}} W(k, k'|\boldsymbol{c}) \frac{\delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})}}{Z(\boldsymbol{X})} \tag{4.1.6}$$

In this chapter we study the leading orders in the system size N of the Shannon entropy per node of the above tailored random graph ensembles, from which the effective number of graphs

with the prescribed distribution $p(X)$ of features follows as $\mathcal{N} = \exp(NS)$

$$\begin{aligned} S &= -\frac{1}{N}\sum_{\boldsymbol{c}} p(\boldsymbol{c})\log p(\boldsymbol{c}) \\ &= -\frac{1}{N}\sum_{\boldsymbol{X}}\frac{\prod_i p(X_i)}{Z(\boldsymbol{X})}\sum_{\boldsymbol{c}}\delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})}\log\Big[\sum_{\boldsymbol{X}'}\frac{\prod_j p(X'_j)}{Z(\boldsymbol{X}')}\delta_{\boldsymbol{X}',\boldsymbol{X}(\boldsymbol{c})}\Big] \\ &= -\frac{1}{N}\sum_{\boldsymbol{X}}\frac{\prod_i p(X_i)}{Z(\boldsymbol{X})}\sum_{\boldsymbol{c}}\delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})}\log\Big[\frac{\prod_j p(X_j)}{Z(\boldsymbol{X})}\Big] \\ &= \sum_{\boldsymbol{X}} p(\boldsymbol{X})S(\boldsymbol{X}) - \sum_{X} p(X)\log p(X) \end{aligned} \tag{4.1.7}$$

with

$$S(\boldsymbol{X}) = \frac{1}{N}\log Z(\boldsymbol{X}) = \frac{1}{N}\log\sum_{\boldsymbol{c}}\delta_{\boldsymbol{X},\boldsymbol{X}(\boldsymbol{c})} \tag{4.1.8}$$

The core of the entropy calculation is determining the leading orders in N of $S(\boldsymbol{X})$, which is the Shannon entropy per node of the ensemble $p(\boldsymbol{c}|\boldsymbol{X})$ in which all node-specific values $\boldsymbol{X} = (X_1, \ldots, X_N)$ are constrained. For $p(X) = p(k)$ this calculation has already been done by Annibale et al. [2009] and chapter 3 (undirected and directed case respectively). For $p(X) = p(k, m)$ it has only partly been done Coolen et al. [2009]. Here we investigate the relation between the entropies of the $p(k)$ and $p(k, m)$ ensembles and the entropy of the ensemble in which the distribution $p(n)$ of local neighbourhoods is imposed.

4.2 Building blocks of the entropy calculations

4.2.1 Relations between feature distributions for nondirected graphs

Since the generalised degrees (k_i, m_i) can be calculated from the local neighbourhoods for any graph $\boldsymbol{c}$, it is clear that the empirical distribution $p(k, m|\boldsymbol{c}) = N^{-1}\sum_i \delta_{k,k_i(\boldsymbol{c})}\delta_{m,m_i(\boldsymbol{c})}$ for any graph can be calculated from the empirical neighbourhood distribution $p(n|\boldsymbol{c}) = N^{-1}\sum_i \delta_{n,n_i(\boldsymbol{c})}$. If we denote with $k(n)$ the central degree k in $n = (k;\ \{\xi^s\})$, we indeed obtain

$$p(k, m|\boldsymbol{c}) = \frac{1}{N}\sum_i \delta_{k,k_i(\boldsymbol{c})}\delta_{m,m_i(\boldsymbol{c})}\sum_n \delta_{n,n_i} = \sum_n p(n)\ \delta_{k,k(n)}\delta_{m,\sum_{s\le k(n)}\xi^s} \tag{4.2.1}$$

Less trivial is the statement that also the distribution $W(k,k'|\boldsymbol{c})$ (as defined in equation 4.1.5) can be written in terms of $p(n|\boldsymbol{c})$. Using $\sum_{ij}c_{ij}=N\bar{k}(\boldsymbol{c})$, with $\bar{k}(\boldsymbol{c})=N^{-1}\sum_i k_i(\boldsymbol{c})$ we obtain

$$W(k,k'|\boldsymbol{c}) = \frac{\sum_i \delta_{k,k_i(\boldsymbol{c})}\sum_{j\in\partial_i}\delta_{k',k_j(\boldsymbol{c})}}{N\sum_n p(n|\boldsymbol{c})k(n)} = \frac{\sum_i\sum_n \delta_{n,n_i(\boldsymbol{c})}\delta_{k,k(n)}\sum_{s\leq k(n)}\delta_{k',\xi^s}}{N\sum_n p(n|\boldsymbol{c})k(n)}$$
$$= \frac{\sum_n p(n|\boldsymbol{c})\delta_{k,k(n)}\sum_{s\leq k(n)}\delta_{k',\xi^s}}{\sum_n p(n|\boldsymbol{c})k(n)} \tag{4.2.2}$$

Given the symmetry of $W(k,k'|\boldsymbol{c})$ under permutation of k and k' we then also have

$$W(k,k'|\boldsymbol{c}) = \frac{\sum_n p(n|\boldsymbol{c})\delta_{k',k(n)}\sum_{s\leq k(n)}\delta_{k,\xi^s}}{\sum_n p(n|\boldsymbol{c})k(n)} \tag{4.2.3}$$

The converse of the above statements is not true. One cannot calculate the neighbourhoud distribution $p(n|\boldsymbol{c})$ from $p(k,m|\boldsymbol{c})$ or from $W(k,k'|\boldsymbol{c})$ (or both). Note that by definition (and since $\boldsymbol{c}$ is nondirected) we always have $W(k,k'|\boldsymbol{c})=W(k',k|\boldsymbol{c})$.

4.2.2 Decomposition of graphs into directed degree-regular subgraphs

Any nondirected graph $\boldsymbol{c}$ can always be decomposed uniquely into a collection of non-overlapping N-node subgraphs $\boldsymbol{\beta}^{kk'}$, with $k,k'\in\mathbb{N}$, which share the nodes $\{1,\ldots,N\}$ of $\boldsymbol{c}$ but not all of the links. These subgraphs are defined for each (k,k') by the adjacency matrices

$$\beta_{ij}^{kk'} = c_{ij}\delta_{k,k_i(\boldsymbol{c})}\delta_{k',k_j(\boldsymbol{c})} \tag{4.2.4}$$

Each graph $\boldsymbol{\beta}^{kk'}$ contains those links in $\boldsymbol{c}$ that go from a node with degree k' to a node with degree k. Clearly, all graphs $\boldsymbol{\beta}^{kk'}$ follow uniquely from $\boldsymbol{c}$ via equation 4.2.4. The converse uniqueness of $\boldsymbol{c}$, given the matrices $\boldsymbol{\beta}^{kk'}$, is a consequence of the simple identity

$$c_{ij} = c_{ij}\sum_{kk'\geq 0}\delta_{k,k_i(\boldsymbol{c})}\delta_{k',k_j(\boldsymbol{c})} = \sum_{kk'\geq 0}\delta_{k,k_i(\boldsymbol{c})}\delta_{k',k_j(\boldsymbol{c})}c_{ij} = \sum_{kk'\geq 0}\beta_{ij}^{kk'} \tag{4.2.5}$$

The graph $\boldsymbol{\beta}^{kk'}$ is directed if $k\neq k'$, and nondirected if $k=k'$. From the symmetry of $\boldsymbol{c}$ it follows moreover that $\beta_{ji}^{kk'}=\beta_{ij}^{k'k}$ for all (i,j,k,k'), so $\boldsymbol{\beta}^{k'k}$ is specified in full by $\boldsymbol{\beta}^{kk'}$. Although each $\boldsymbol{\beta}^{kk'}$ is an N-node graph, most of the nodes in $\boldsymbol{\beta}^{kk'}$ will be isolated: all nodes whose degrees in the original graph $\boldsymbol{c}$ were neither k nor k' will have degree zero in $\boldsymbol{\beta}^{kk'}$.

We now inspect the degree statistics of the decomposition graphs $\boldsymbol{\beta}^{kk'}$, and their relation with the structural features of $\boldsymbol{c}$. If $k\neq k'$ we find for the remaining degrees in $\boldsymbol{\beta}^{kk'}$

$$k_i(\boldsymbol{c})=k: \qquad k_i^{\rm in}(\boldsymbol{\beta}^{kk'})=\sum_{j\in\partial_i}\delta_{k',k_j(\boldsymbol{c})}, \quad k_i^{\rm out}(\boldsymbol{\beta}^{kk'})=0 \tag{4.2.6}$$

$$k_j(\boldsymbol{c})=k': \qquad k_j^{\rm out}(\boldsymbol{\beta}^{kk'})=\sum_{i\in\partial_j}\delta_{k,k_i(\boldsymbol{c})}, \quad k_j^{\rm in}(\boldsymbol{\beta}^{kk'})=0 \tag{4.2.7}$$

Hence the joint in-out degree distribution of $\boldsymbol{\beta}^{kk'}$ can be writen in terms of the empirical distribution of neighbourhoods of $\boldsymbol{c}$, viz. $p(n|\boldsymbol{c}) = N^{-1}\sum_i \delta_{n,n_i(\boldsymbol{c})}$ with $n = (k;\ \{\xi^s\})$

$$\begin{aligned}
p^{kk'}(q^{\rm in},q^{\rm out}) &= \frac{1}{N}\sum_i \delta_{q^{\rm in},k_i^{\rm in}(\boldsymbol{\beta}^{kk'})}\,\delta_{q^{\rm out},k_i^{\rm out}(\boldsymbol{\beta}^{kk'})} \\
&= \frac{1}{N}\sum_i \delta_{q^{\rm in},\delta_{k,k_i(\boldsymbol{c})}\sum_{j\in\partial_i}\delta_{k',k_j(\boldsymbol{c})}}\,\delta_{q^{\rm out},\delta_{k',k_i(\boldsymbol{c})}\sum_{j\in\partial_i}\delta_{k,k_j(\boldsymbol{c})}} \\
&= \frac{1}{N}\sum_i \Big[\delta_{k,k_i(\boldsymbol{c})}\delta_{q^{\rm in},\sum_{j\in\partial_i}\delta_{k',k_j(\boldsymbol{c})}} + (1-\delta_{k,k_i(\boldsymbol{c})})\delta_{q^{\rm in},0}\Big] \\
&\qquad\times\Big[\delta_{k',k_i(\boldsymbol{c})}\delta_{q^{\rm out},\sum_{j\in\partial_i}\delta_{k,k_j(\boldsymbol{c})}} + (1-\delta_{k',k_i(\boldsymbol{c})})\delta_{q^{\rm out},0}\Big] \\
&= \sum_n p(n|\boldsymbol{c})\Big[\delta_{k,k(n)}\delta_{q^{\rm in},\sum_{s\le k(n)}\delta_{k',\xi^s(n)}} + (1-\delta_{k,k(n)})\delta_{q^{\rm in},0}\Big] \\
&\qquad\times\Big[\delta_{k',k(n)}\delta_{q^{\rm out},\sum_{s\le k(n)}\delta_{k,\xi^s(n)}} + (1-\delta_{k',k(n)})\delta_{q^{\rm out},0}\Big] \qquad (4.2.8)
\end{aligned}$$

The two marginals of 4.2.8 are

$$p_{\rm in}^{kk'}(q) = \sum_n p(n|\boldsymbol{c})\Big[\delta_{k,k(n)}\delta_{q,\sum_{s\le k(n)}\delta_{k',\xi^s(n)}} + (1-\delta_{k,k(n)})\delta_{q,0}\Big] \qquad (4.2.9)$$

$$p_{\rm out}^{kk'}(q) = \sum_n p(n|\boldsymbol{c})\Big[\delta_{k',k(n)}\delta_{q,\sum_{s\le k(n)}\delta_{k,\xi^s(n)}} + (1-\delta_{k',k(n)})\delta_{q,0}\Big] \qquad (4.2.10)$$

Hence $p_{\rm in}^{kk'}(q) = p_{\rm out}^{k'k}(q)$, as expected. The average degree $\bar{q}^{kk'} = \sum_{q^{\rm in},q^{\rm out}} q^{\rm in}p^{kk'}(q^{\rm in},q^{\rm out}) = \sum_{q^{\rm in},q^{\rm out}} q^{\rm out}p^{kk'}(q^{\rm in},q^{\rm out})$ of the graph $\boldsymbol{\beta}^{kk'}$ can be written, using identity 4.2.3 and the symmetry of $W(k,k'|\boldsymbol{c})$, as

$$\bar{q}^{kk'} = \sum_n p(n|\boldsymbol{c})\delta_{k(n),k}\sum_{s\le k(n)}\delta_{k',\xi^s(n)} = \bar{k}(\boldsymbol{c})W(k,k'|\boldsymbol{c}) \qquad (4.2.11)$$

If $k = k'$, the decomposition matrix $\boldsymbol{\beta}^{kk'}$ is symmetric. Here we find

$$k_i(\boldsymbol{\beta}^{kk}) = \delta_{k,k_i(\boldsymbol{c})}\sum_{j\in\partial_i}\delta_{k,k_j(\boldsymbol{c})} \qquad (4.2.12)$$

Hence the degree distribution of $\boldsymbol{\beta}^{kk}$ becomes

$$\begin{aligned}
p^{kk}(q) &= \frac{1}{N}\sum_i \delta_{q,\delta_{k,k_i(\boldsymbol{c})}\sum_{j\in\partial_i}\delta_{k,k_j(\boldsymbol{c})}} \\
&= \frac{1}{N}\sum_i \Big[\delta_{k,k_i(\boldsymbol{c})}\delta_{q,\sum_{j\in\partial_i}\delta_{k,k_j(\boldsymbol{c})}} + (1-\delta_{k,k_i(\boldsymbol{c})})\delta_{q,0}\Big] \\
&= \sum_n p(n|\boldsymbol{c})\Big[\delta_{k,k(n)}\delta_{q,\sum_{s\le k(n)}\delta_{k,\xi^s(n)}} + (1-\delta_{k,k(n)})\delta_{q,0}\Big] \qquad (4.2.13)
\end{aligned}$$

The average degree in $\boldsymbol{\beta}^{kk}$ is therefore

$$\bar{q}^{kk} = \sum_n p(n|\boldsymbol{c})\delta_{k(n),k}\sum_{s\le k(n)}\delta_{k,\xi^s(n)} = \bar{k}(\boldsymbol{c})W(k,k|\boldsymbol{c}) \qquad (4.2.14)$$

4.3 Entropy of ensembles of bipartite graphs

Here we calculate the leading orders in N of the entropy per node 4.1.7 for ensembles of bipartite graphs with prescribed (and possibly distinct) degree distributions in the two node sets. This is not only a novel result in itself, but will also form the seed of the entropy calculation for ensembles with constrained neighbourhoods in a subsequent section.

In a bipartite ensemble the N nodes can be divided into two disjoint sets $A, B \subseteq \{1, \ldots, N\}$ such that $c_{ij} = 0$ if $i, j \in A$ or $i, j \in B$, leaving only links *between* A and B. This allows us to draw upon results on directed graphs derived in chapter 3. The directed graph $\boldsymbol{c}'$ associated with the bipartite graph $\boldsymbol{c}$ would have

$$j \in B \text{ or } i \in A: \qquad c'_{ij} = 0 \tag{4.3.1}$$

$$j \in A \text{ and } i \in B: \qquad c'_{ij} = c_{ij} \tag{4.3.2}$$

and hence the in- and out-degree sequence $\vec{\boldsymbol{k}} = ((k_1^{\text{in}}, k_1^{\text{out}}), \ldots, (k_N^{\text{in}}, k_N^{\text{out}}))$ of $\boldsymbol{c}'$ can be expressed in terms of the degree sequence $\boldsymbol{k}$ of $\boldsymbol{c}$ via

$$i \in A: \qquad \vec{k}_i = (k_i^{\text{in}}, k_i^{\text{out}}) = (0, k_i) \tag{4.3.3}$$

$$i \in B: \qquad \vec{k}_i = (k_i^{\text{in}}, k_i^{\text{out}}) = (k_i, 0) \tag{4.3.4}$$

The directed graph will thus have the joint degree distribution

$$p(q^{\text{in}}, q^{\text{out}}) = \frac{|A|}{N}\delta_{q^{\text{in}},0}p_A(q^{\text{out}}) + (1-\frac{|A|}{N})p_B(q^{\text{in}})\delta_{q^{\text{out}},0} \tag{4.3.5}$$

with the degree distributions $p_A(k) = |A|^{-1}\sum_{i\in A}\delta_{k,k_i(\boldsymbol{c})}$ and $p_B(k) = |B|^{-1}\sum_{i\in B}\delta_{k,k_i(\boldsymbol{c})}$ in the sets A and B of the bipartite graph. Our bipartite ensemble is one in which we describe the distributions $p_A(k)$ and $p_B(k)$, together with the probability $f \in [0, 1]$ for a node to be in subset A, and we forbid links within the sets A or B. Conservation of links demands that the two distributions cannot be independent, but must obey $\bar{q} = (1-f)\sum_q q p_B(q) = f\sum_q q p_A(q)$, where $\bar{q}$ is the average degree. Our bijective mapping to directed graphs shows that the entropy of any bipartite ensemble can be calculated by application of equation 4.1.7,4.1.8 to an ensemble of directed graphs, with $X_i = (\tau_i, k_i)$. Here $\tau_i \in \{A, B\}$ gives the subset assignment of a node. We then find

$$\begin{aligned} S &= \sum_{\boldsymbol{\tau},\boldsymbol{k}}\Big[\prod_i p(\tau_i, k_i)\Big]S(\boldsymbol{\tau}, \boldsymbol{k}) - f\log f - (1-f)\log(1-f) \\ &\quad -f\sum_k p_A(k)\log p_A(k) - (1-f)\sum_k p_B(k)\log p_B(k) \end{aligned} \tag{4.3.6}$$

with

$$p(\tau,k) = f\delta_{\tau,A}p_A(k) + (1-f)\delta_{\tau,B}p_B(k) \tag{4.3.7}$$

$$S(\boldsymbol{\tau},\boldsymbol{k}) = \frac{1}{N}\log\sum_{\boldsymbol{c}}\Big(\prod_{i,\tau_i=A}\delta_{\vec{k}_i,(0,k_i)}\Big)\Big(\prod_{i,\tau_i=B}\delta_{\vec{k}_i,(k_i,0)}\Big) \tag{4.3.8}$$

The latter quantity follows from the calculation in 3, with the short-hand $\pi_{\bar{q}}(q) = \mathrm{e}^{-\bar{q}}\bar{q}^q/q!$ and modulo terms that vanish for $N\to\infty$:

$$\begin{aligned} S(\boldsymbol{\tau},\boldsymbol{k}) &= \bar{q}[\log(N/\bar{q})+1] + \sum_q\Big[f\delta_{q,0} + (1-f)p_B(q)\Big]\log\pi_{\bar{q}}(q) \\ &\quad + \sum_q\Big[fp_A(q) + (1-f)\delta_{q,0}\Big]\log\pi_{\bar{q}}(q) \\ &= \bar{q}\log(N/\bar{q}) + f\sum_q p_A(q)\log\pi_{\bar{q}}(q) + (1-f)\sum_q p_B(q)\log\pi_{\bar{q}}(q) \end{aligned} \tag{4.3.9}$$

This then leads to our final result for the entropy per node of tailored bipartite graph ensembles, with imposed bipartite degree distributions $p_A(k)$ and $p_B(k)$, average degree $\bar{k}$, and a fraction f of nodes in the set A (modulo vanishing orders in N):

$$\begin{aligned} S &= \bar{k}\log(N/\bar{k}) - f\log f - (1-f)\log(1-f) \\ &\quad -f\sum_k p_A(k)\log\Big(\frac{p_A(k)}{\pi_{\bar{k}}(k)}\Big) - (1-f)\sum_k p_B(k)\log\Big(\frac{p_B(k)}{\pi_{\bar{k}}(k)}\Big) \end{aligned} \tag{4.3.10}$$

If the sets A and B were to be specified explicitly (as opposed to only their relative sizes), the contribution $S_f = -f\log f - (1-f)\log(1-f)$ would disappear from the above formula.

4.4 Entropy of ensembles with constrained neighbourhoods

We now turn to the Shannon entropy per node (equation 4.1.7) of the ensemble given by 4.1.2 in which for the observables $X_i(\boldsymbol{c})$ we choose the local neighbourhood $n_i(\boldsymbol{c})$ defined in equation 4.1.1. For this we need to calculate the leading orders of $S(\boldsymbol{n}) = N^{-1}\log\sum_{\boldsymbol{c}}\delta_{\boldsymbol{n},\boldsymbol{n}(\boldsymbol{c})}$. We now use the one-to-one relationship between a graph $\boldsymbol{c}$ and its decomposition $\boldsymbol{c} = \sum_{qq'}\boldsymbol{\beta}^{qq'}$, to write

$$S(\boldsymbol{n}) = \frac{1}{N}\log\sum_{\{\boldsymbol{\beta}^{kk'}\}}\delta_{\boldsymbol{n},\boldsymbol{n}(\boldsymbol{c})} \tag{4.4.1}$$

The next argument is the key to our ability to evaluate the entropy. It involves translating the constraint $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{c})$ into constraints on the decomposition matrices $\boldsymbol{\beta}^{kk'}$. Let us define the sets of nodes in $\boldsymbol{c}$ which have the same degree, viz. $I_k(\boldsymbol{n}) = \{i\le N|\, k_i(\boldsymbol{c}) = k\}$. The constraint $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{c})$ in equation 4.4.1 prescribes

(*i*) all the sets I_k of nodes with a given degree

(*ii*) for each node $i \in I_k$ which sets $I_{k'}$ this node is (possibly multiply) connected to

Hence the constraint $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{c})$ specifies exactly the in- and out-degree sequences of all decomposition matrices $\boldsymbol{\beta}^{kk'}$ of $\boldsymbol{c}$, which we will denote as $\vec{\boldsymbol{q}}^{kk'} = (\boldsymbol{q}^{\text{in},kk'}, \boldsymbol{q}^{\text{out},kk'})$, and whose distributions we have already calculated in equations 4.2.8 and 4.2.13. We thus see that equation 4.4.1 can be written as

$$S(\boldsymbol{n}) = \frac{1}{N}\log \sum_{\{\boldsymbol{\beta}^{kk'}\}} \prod_{kk'} \delta_{\vec{\boldsymbol{q}}_{\boldsymbol{n}}^{kk'},\vec{\boldsymbol{q}}(\boldsymbol{\beta}^{kk'})} \tag{4.4.2}$$

in which $\vec{\boldsymbol{q}}_{\boldsymbol{n}}^{kk'}$ are the in- and out-degree sequences that are imposed by the local environment sequence $\boldsymbol{n}$ on the decomposition matrix $\boldsymbol{\beta}^{kk'}$, and whose distributions are known to be as given by equations 4.2.8,4.2.13. Using the symmetry $(\boldsymbol{\beta}^{kk'})^{\dagger} = \boldsymbol{\beta}^{k'k}$ we may now write

$$\begin{aligned} S(\boldsymbol{n}) &= \frac{1}{N}\log\Big[\Big(\prod_{k<k'}\sum_{\boldsymbol{\beta}^{kk'}} \delta_{\vec{\boldsymbol{q}}_{\boldsymbol{n}}^{kk'},\vec{\boldsymbol{q}}(\boldsymbol{\beta}^{kk'})}\Big)\Big(\prod_{k}\sum_{\boldsymbol{\beta}^{kk}} \delta_{\vec{\boldsymbol{q}}_{\boldsymbol{n}}^{kk},\vec{\boldsymbol{q}}(\boldsymbol{\beta}^{kk})}\Big)\Big] \\ &= \sum_{k<k'}\Big\{\frac{1}{N}\log\sum_{\boldsymbol{\beta}^{kk'}} \delta_{\vec{\boldsymbol{q}}_{\boldsymbol{n}}^{kk'},\vec{\boldsymbol{q}}(\boldsymbol{\beta}^{kk'})}\Big\} + \sum_{k}\Big\{\frac{1}{N}\log\sum_{\boldsymbol{\beta}^{kk}} \delta_{\boldsymbol{q}_{\boldsymbol{n}}^{kk},\boldsymbol{q}(\boldsymbol{\beta}^{kk})}\Big\} \end{aligned} \tag{4.4.3}$$

We see that the entropy $S(\boldsymbol{n})$ can be written as the sum of the entropies of sub-ensembles, which are the decomposition matrices $\boldsymbol{\beta}^{kk'}$ with prescribed degree sequences. The second sum in equation 4.4.3 is over nondirected ensembles, the first over directed ones. The sub-entropies were all calculated, respectively, in Annibale et al. [2009] and chapter 3 [1]. The entropy of an N-node nondirected random graph ensemble with degree sequence $\boldsymbol{q}$ was found to be (modulo terms that vanish for $N \to \infty$):

$$S_{\boldsymbol{q}} = \frac{1}{N}\log\sum_{\boldsymbol{c}} \delta_{\boldsymbol{q},\boldsymbol{q}(\boldsymbol{c})} = \frac{1}{2}\bar{q}[\log(N/\bar{q}) + 1] + \sum_{q} p(q)\log\pi_{\bar{q}}(q) \tag{4.4.4}$$

in which $\bar{q} = N^{-1}\sum_i q_i$ and $\pi_{\bar{q}}(q)$ is the Poisson distribution with average $\bar{q}$. The entropy of an N-node directed random graph ensemble with in- and out-degree sequence $\vec{\boldsymbol{q}}$ was found to be (modulo terms that vanish for $N \to \infty$):

$$\begin{aligned} S_{\vec{\boldsymbol{q}}} &= \frac{1}{N}\log\sum_{\boldsymbol{c}} \delta_{\vec{\boldsymbol{q}},\vec{\boldsymbol{q}}(\boldsymbol{c})} \\ &= \bar{q}[\log(N/\bar{q}) + 1] + \sum_{q^{\text{in}},q^{\text{out}}} p(q^{\text{in}},q^{\text{out}})\log[\pi_{\bar{q}}(q^{\text{in}})\pi_{\bar{q}}(q^{\text{out}})] \end{aligned} \tag{4.4.5}$$

[1] In chapter 3 the entropies were carried out for ensembles with prescribed degree distributions, but it was shown that, in analogy with equation 4.1.7, this is simply the sum of the Shannon entropy of the degree distributions and the entropy of the corresponding ensemble with prescribed sequences.

The above entropies depend in leading orders only on the degree distributions (as opposed to the degree sequences), and since these distributions were already calculated in equation 4.2.8,4.2.13, we can simply insert expressions 4.4.4,4.4.5 into equation 4.4.3, with the correct distributions as given by expression 4.2.8,4.2.13, and find an expression that depends only on the local environment distribution $p(n) = N^{-1}\sum_i \delta_{n,n_i}$:

$$\begin{aligned}
S(\boldsymbol{n}) &= \sum_{k<k'}\Big\{\bar{q}^{kk'}[\log(N/\bar{q}^{kk'})+1] + \sum_{q^{\rm in},q^{\rm out}} p^{kk'}(q^{\rm in},q^{\rm out})\log[\pi_{\bar{q}^{kk'}}(q^{\rm in})\pi_{\bar{q}^{kk'}}(q^{\rm out})]\Big\} \\
&\quad + \sum_k\Big\{\frac{1}{2}\bar{q}^{kk}[\log(N/\bar{q}^{kk})+1] + \sum_q p^{kk}(q)\log\pi_{\bar{q}^{kk}}(q)\Big\} \\
&= \frac{1}{2}\sum_{k\neq k'}\Big\{\bar{k}W(k,k')[\log(N/\bar{k}W(k,k'))-1] \\
&\quad +2\bar{k}W(k,k')\log[\bar{k}W(k,k')] - \sum_q[p^{kk'}_{\rm in}(q)+p^{kk'}_{\rm out}(q)]\log q!\Big\} \\
&\quad +\frac{1}{2}\sum_k\Big\{\bar{k}W(k,k)[\log(N/\bar{k}W(k,k))-1] \\
&\quad +2\bar{k}W(k,k)\log[\bar{k}W(k,k)] - 2\sum_q p^{kk}(q)\log q!\Big\} \\
&= \frac{1}{2}\bar{k}[\log(N/\bar{k})-1] + \frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k') \\
&\quad -\sum_q\Big[\frac{1}{2}\sum_{k\neq k'}[p^{kk'}_{\rm in}(q)+p^{kk'}_{\rm out}(q)] + \sum_k p^{kk}(q)\Big]\log q! \\
&= \frac{1}{2}\bar{k}[\log(N/\bar{k})-1] + \frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k') \\
&\quad -\sum_q\sum_n p(n)\sum_{k,k'}\Big[\delta_{k,k(n)}\delta_{q,\sum_{s\leq k(n)}\delta_{k',\xi^s(n)}} + (1-\delta_{k',k(n)})\delta_{q,0}\Big]\log q! \\
&= \frac{1}{2}\bar{k}[\log(N/\bar{k})-1] + \frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k') \\
&\quad -\sum_n p(n)\sum_k\log\Big[\Big(\sum_{s\leq k(n)}\delta_{k,\xi^s(n)}\Big)!\Big]
\end{aligned} \tag{4.4.6}$$

Insertion of this result into the general formula 4.1.7 gives us an analytical expression for the Shannon entropy of the random graph ensemble with prescribed distribution $p(n)$ of local neighbourhoods, modulo terms that vanish for $N\to\infty$. This expression is fully explicit, since $\bar{k}$ and $W(k,k')$ are both determined by the distribution $p(n)$, via $\bar{k} = \sum_n p(n)k(n)$ and equation 4.2.3 respectively:

$$\begin{aligned}
S &= \frac{1}{2}\bar{k}[\log(N/\bar{k})-1] + \frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k') \\
&\quad -\sum_n p(n)\log p(n) - \sum_n p(n)\sum_k\log\Big[\Big(\sum_{s\leq k(n)}\delta_{k,\xi^s(n)}\Big)!\Big]
\end{aligned} \tag{4.4.7}$$

4.5 Entropy of ensembles of networks with specified generalized degree distribution

In this section we consider an ensemble of nondirected networks with a specified generalized degree distribution $p(k,m) = N^{-1}\sum_i \delta_{k,k_i(\boldsymbol{c})}\delta_{m,m_i(\boldsymbol{c})}$, where $k_i(\boldsymbol{c}) = \sum_j c_{ij}$ and $m_i = \sum_{jk} c_{ij}c_{jk}$. Previous work by Coolen et al. [2009] began this calculation, and reached (in leading order) the intermediate form set out below

$$\begin{aligned} S &= \frac{1}{2}\bar{k}[\log(N/\bar{k})+1] - \sum_{k,m} p(k,m)\log\Big(\frac{p(k,m)}{\pi(k)}\Big) \\ &\quad + \sum_{k,m} p(k,m)\log\Big(\sum_{\xi^1,..,\xi^k}\delta_{m,\sum_{s=1}^k \xi^s}\prod_{s=1}^{k}\gamma(k,\xi^s)\Big) \end{aligned} \tag{4.5.1}$$

$\bar{k}$ indicates the average degree; $\pi_{\bar{k}}(k)$ is the Poissonian distribution with average degree $\bar{k}$. The sum inside the logarithm in the final term of 4.5.1 runs over all sets of k nonnegative integers $\xi^1 \ldots \xi^k$. The function $\gamma(.,.)$ is defined as the non-negative solution to the following self-consistency relation

$$\gamma(k,k') = \sum_{m'}\frac{k'}{\bar{k}}\,p(k',m')\left[\frac{\sum_{\xi^1\ldots\xi^{k'-1}}\delta_{m'-k,\sum_{s=1}^{k'-1}\xi^s}\prod_{s=1}^{k'-1}\gamma\,(k',\xi^s)}{\sum_{\xi^1\ldots\xi^{k'}}\delta_{m',\sum_{s=1}^{k'}\xi^s}\prod_{s=1}^{k'}\gamma\,(k',\xi^s)}\right] \tag{4.5.2}$$

This equation does not yield to a straightforward solution, and can only be evaluated numerically or in certain special cases. Without a physical interpretation of $\gamma(k,k')$, this intermediate answer is limited in how much insight it can provide. We will now show how the entropy can be expressed in terms of measurable quantities.

Our strategy is to derive an expression for the (observable) degree-degree correlations $W(k,k')$, and show that these can be expressed in terms of the order parameter $\gamma(k,k')$ that appears in equation 4.5.1. We calculate the average of this quantity in our tailored ensembles of the form equation 4.1.2, where we now define topological characteristics by specifying a generalised degree distribution $p(k,m)$. We follow closely the steps taken in Coolen et al. [2009] and write,

for the ensemble with a specified generalised degree sequence $(\boldsymbol{k}, \boldsymbol{m})$

$$
\begin{aligned}
W(k,k') =& \frac{1}{N\bar{k}} \sum_{\boldsymbol{c}} p(\boldsymbol{c}|\boldsymbol{k},\boldsymbol{m}) \sum_{rs} c_{rs}\delta_{k,\sum_\ell c_{r\ell}}\delta_{k',\sum_\ell c_{s\ell}} \\
=& \frac{1}{N^2} \sum_{rs} \delta_{k,k_r}\delta_{k',k_s} \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})-\mathrm{i}(\theta_r+\theta_s+\phi_r k_s+\phi_s k_r)} \prod_{i<j}\left[1+\frac{\bar{k}}{N}\left(\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}-1\right)\right]}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})} \prod_{i<j}\left[1+\frac{\bar{k}}{N}\left(e^{-i(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}-1\right)\right]} + O(\frac{1}{N}) \\
=& \frac{1}{N^2} \sum_{rs} \delta_{k,k_r}\delta_{k',k_s} \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})-\mathrm{i}(\theta_r+\theta_s+\phi_r k'+\phi_s k)+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}+\ldots}}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}+\ldots}} + O(N^{-1}) \\
=& \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}\left(\frac{1}{N^2}\sum_{rs}\delta_{k,k_r}\delta_{k',k_s}\mathrm{e}^{-\mathrm{i}(\theta_r+\theta_s+\phi_r k'+\phi_s k)}\right)}}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}} + O(N^{-1}) \\
=& \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}\left(\frac{1}{N}\sum_r \delta_{k,k_r}\mathrm{e}^{-\mathrm{i}(\theta_r+\phi_r k')}\right)\left(\frac{1}{N}\sum_s \delta_{k',k_s}\mathrm{e}^{-\mathrm{i}(\theta_s+\phi_s k)}\right)}}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}} + O(N^{-1}) \\
=& \frac{\int\{\mathrm{d}P\mathrm{d}\hat{P}\}\mathrm{e}^{N\Psi[P,\hat{P}]}\left(\int \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k)\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k'}\right)\left(\int \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k')\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k}\right)}{\int\{\mathrm{d}P\mathrm{d}\hat{P}\}\mathrm{e}^{N\Psi[P,\hat{P}]}} + O(N^{-1})
\end{aligned}
\tag{4.5.3}
$$

Taking the limit $N \to \infty$ therefore gives

$$
\lim_{N\to\infty} W(k,k') \quad = \quad \left(\int \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k)\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k'}\right)\left(\int \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k')\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k}\right)\Big|_{\text{saddle-point } \{P,\hat{P}\} \text{ of } \Psi} \tag{4.5.4}
$$

in which the function $\Psi[P,\hat{P}]$ is identical to that found in Coolen et al. [2009]. Using the the formulae in Coolen et al. [2009] that relate to the definition of the order parameter $\gamma(k,k')$, we then obtain for $N \to \infty$ the unexpected simple but welcome relation

$$
W(k,k') \quad = \quad \gamma(k,k')\gamma(k',k) \tag{4.5.5}
$$

A similar, although slightly more involved, calculation leads to an expression for the joint distribution $W(k,m;k',m')$; see Appendix B for details.

Our final aim is to use identity 4.5.5 to resolve equation 4.5.1 into observable quantities. Consider the nontrivial term in equation 4.5.1

$$
\Gamma \quad = \quad \sum_{k,m} p(k,m) \log\Big(\sum_{\xi^1,..,\xi^k} \prod_{s=1}^{k} \gamma(k,\xi^s)\delta_{m,\sum_{s=1}^k \xi^s}\Big) \tag{4.5.6}
$$

At this point of the calculation, the effect of factorising across nodes has been to break the expression down into terms which, for every generalised degree (k,m), enumerate all the possible ways of dividing m second neighbours between k first neighbours. The term inside the logarithm sums for each k over all configurations $\{\xi^1 \ldots \xi^k\}$ which meet the condition $\sum_{s=1}^k \xi^s = m$. To formalise this idea, we may re-aggregate the expression for any graphically realisable distribution

$p(k,m)$ to write

$$\begin{aligned}\Gamma &= \frac{1}{N}\log\prod_{k,m}\Big(\sum_{\xi^1,..,\xi^k}\prod_{s=1}^{k}\gamma(k,\xi^s)\delta_{m,\sum_{s=1}^{k}\xi^s}\Big)^{Np(k,m)}\\ &= \frac{1}{N}\log\prod_{i}\Big(\sum_{\xi_i^1...\xi_i^{k_i}}\Big[\prod_{s=1}^{k_i}\gamma(k_i,\xi_i^s)\Big]\delta_{m_i,\sum_{s=1}^{k_i}\xi_i^s}\Big)\\ &= \frac{1}{N}\log\Big\{\sum_{\xi_1^1...\xi_1^{k_1}}\cdots\sum_{\xi_N^1...\xi_N^{k_N}}\Big(\prod_i\delta_{m_i,\sum_{s=1}^{k_i}\xi_i^s}\Big)\prod_i\prod_{s=1}^{k_i}\gamma(k_i,\xi_i^s)\Big\} \qquad (4.5.7)\end{aligned}$$

We can now see that the separate terms precisely enumerate all the permutations of degrees and neighbour-degrees for networks with a generalised degree sequence consistent with any pair (k,m) appearing $Np(k,m)$ times. The Kronecker deltas $\delta_{m_i,\sum_{s=1}^{k_i}\xi_i^s}$ tell us that each ξ_i^s in any nonzero term is to be interpreted as the degree of a node $j\in\partial_i$, and must therefore appear also as the left argument in another factor of the type $\gamma(k_j,.)$. This insight allows the expression to be substantially simplified, since we already know that $\gamma(k,k')\gamma(k',k) = W(k',k)$ where $W(k',k)$ is the correlation between degrees of connected nodes. Hence, any nonvanishing contribution to the sum over all neighbourhoods inside the logarithm of 4.5.7 will be equal to a repeated product of factors $W(k,k')$, with different (k,k'). Since we also know that the number of links between nodes with degree combination (k,k') equals $N\bar{k}W(k,k')$ in leading order in N, we conjecture that in leading order we may make the following replacement inside 4.5.7

$$\prod_i\prod_{s=1}^{k_i}\gamma(k_i,\xi_i^s) \quad\to\quad \prod_{k,k'}W(k,k')^{\frac{\bar{k}N}{2}W(k,k')} \qquad (4.5.8)$$

(where the factor $\frac{1}{2}$ in the exponent reflects the fact that two $\gamma(.,.)$ factors combine to form each factor $W(.,.)$). With this conjecture we obtain, in leading order in N:

$$\begin{aligned}\Gamma =& \frac{1}{N}\log\Big\{\sum_{\xi_1^1...\xi_1^{k_1}}\cdots\sum_{\xi_N^1...\xi_N^{k_N}}\Big(\prod_i\delta_{m_i,\sum_{s=1}^{k_i}\xi_i^s}\Big)\Big\}+\frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k')\\ =& \sum_{k,m}p(k,m)\log\Big(\sum_{\xi^1,..,\xi^k}\delta_{m,\sum_{s=1}^{k}\xi^s}\Big)+\frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k') \qquad (4.5.9)\end{aligned}$$

This implies that equation 4.5.1 simplifies to

$$\begin{aligned}S &= \frac{1}{2}\bar{k}[\log(N/\bar{k})+1]+\frac{1}{2}\bar{k}\sum_{k,k'}W(k,k')\log W(k,k')\\ &\quad -\sum_{k,m}p(k,m)\log\Big(\frac{p(k,m)}{\pi(k)}\Big)+\sum_{k,m}p(k,m)\log\Big(\sum_{\xi^1,..,\xi^k}\delta_{m,\sum_{s=1}^{k}\xi^s}\Big) \qquad (4.5.10)\end{aligned}$$

4.6 Summary

In this chapter we have derived, in leading orders in N, explicit expressions for the Shannon entropies of different types of tailored random graph ensembles, for which no such expressions had yet been obtained. This work builds on and extends the ideas and techniques developed in the three papers Annibale et al. [2009], Coolen et al. [2009], Roberts et al. [2011], which use path integral representations to achieve link factorisation in the various summations over graphs. We show in this chapter how the new ensemble entropies can often be calculated by efficient use and combination of earlier results.

The first class of graph ensembles we studied consists of bipartite nondirected graphs with prescribed (and possibly nonidentical) distributions of degrees for the two node subsets. This case is handled by a bijective mapping from bipartite to directed graphs, for which formulae are available. The second class consists of graphs with prescribed distributions of local neighbourhoods, where the neighbourhood of a node is defined as its own degree plus the values of the degrees of its immediate neighbours. This problem was solved using a decomposition in terms of bipartite graphs, building on the previous result. The final class of graphs, for which the entropy had in the past only partially been calculated, consist of graphs with presecribed distributions of generalised degrees, i.e. of ordinary degrees plus the total number of length-two paths starting in the specified nodes. Here we derive two novel and exact identities linking the order parameters to macroscopic observables, which lead to an explicit entropy formula based on a plausible but not yet proven conjecture,

Since completing this work, our attention has been drawn to a preprint Bordenave and Caputo [2013] which considers the question of the entropy of random graph ensembles constrained with a given distribution of neighbourhoods by a probability theory route, via an adapted Configuration Model. In that case, the neighbourhoods were specified as graphlets of an arbitrary depth. Bordenave and Caputo [2013] also retrieves the entropy of an ensemble constrained with a specified degree distribution, as originally derived by Annibale et al. [2009].

CHAPTER 5

RANDOM GRAPH ENSEMBLES WITH MANY SHORT LOOPS

This chapter will present some contrasting strategies to studying ensembles of networks with controlled numbers of short loops. Section 5.1 begins by providing some background to motivate the problem. It then reviews some interesting results from the literature, illustrating some of the innovative techniques have been employed to circumvent the challenge of not being able to factorise across nodes, which was a key step employed in the previous two chapters. These calculations are qualitatively different from the constraints studied in the previous two sections, because it is no longer valid to assume that the networks in these ensembles are effectively locally tree-like.

Section 5.2 introduces the Strauss Model, which is an exponential random graph model which defines the maximum entropy ensemble for a given average connectivity and average number of triangles. The authors of Burda et al. [2004b] demonstrated how this model can be approached with a perturbative expansion. We then carry out a simple extension of the results of Burda et al. [2004b] to derive the entropy of such an ensemble. Unfortunately, the Strauss model is known to have a critical point, beyond which it condenses into an un-physical 'clumpy' phase. This happens at roughly the parameters which equate to 'one triangle per node'. We show that,

while this is a substantial improvement on the clustering exibited by the Erdös-Rényi model, it is nonetheless well below the clustering values observed in real biological networks.

Section 5.3 takes inspiration from immune-system models, as developed by Agliari et al. [2013a,b]. Here the underlying model is a bipartite graph - which is then projected onto a graph on the nodes from one side of the original bipartite graph. In the projected graph, two nodes are connected if they have a common neighbour in the bipartite graph. This creates a network with a substantial number of loops and cliques. The parameters of the model can be controlled to create the required average number of triangles. The drawback of this model is that it has its own very characteristic topology, where a loop of length k will almost certainly appear within a clique of size k. However, the model is intuitively appealing when applied to protein interaction networks, because the underlying bi-partite graph can be interpreted as showing the protein complexes on the left hand side, and their constituent proteins on the right hand side. This makes this model a promising choice for modelling the clustering in protein interaction networks.

5.1 Motivation and background

In this chapter we will briefly review the frontier of the analytical techniques available to model and rigorously analyse ensembles of random graphs with non-trivial numbers of short loops. Tailored random graph ensembles provide a framework within which we can better understand and quantify topological patterns observed in real life networks. Most analytical approaches assume that the networks under study are locally tree-like. This permits, after appropriate mathematical manipulations, factorisation across nodes, which substantially simplifies averaging operations across the network. However, real-life networks - for example protein-protein interaction networks (PPIN) or social networks - have a significant number of short loops. Figure 5.1 shows a typical trajectory of the average number of loops in a real biological network, compared to its degree randomised counterparts.

It is widely accepted that the abundance of loops in, for example, PPIN is intrinsic to the function of the network. Prill et al. [2005] suggested that the stability of a biological network is highly correlated with the relative abundance of motifs (e.g. triangles) in the network. The authors of Jeong and Berman [2008] and El-Samad et al. [2005] observed an apparent relation-

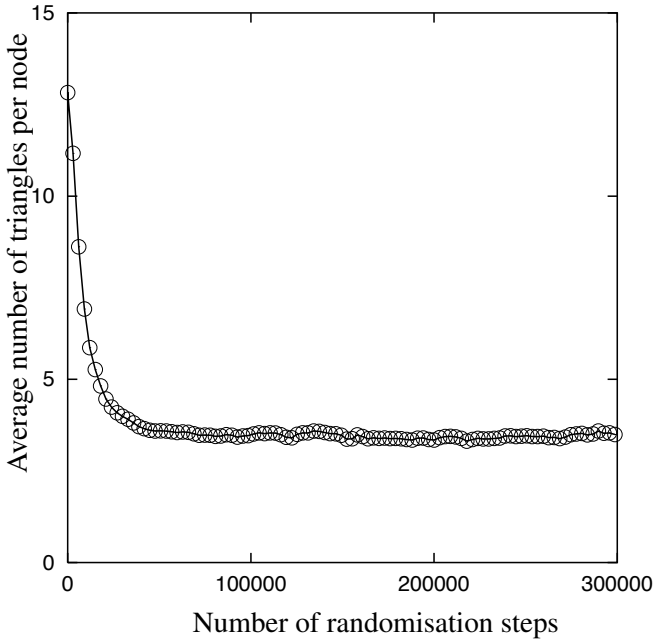


Figure 5.1: Trajectory of average number of triangles per node as a real network undergoes degree-preserving randomisation.

ship between short cycles in gene-regulation networks and the system's response to stress and heat-shock. A highly cited paper Milo et al. [2002] went as far as to propose that motifs (e.g. triangles) are the basic building blocks of most networks. It is evident that incorporating constraints relating to the density and distribution of short loops into the specifications of random graph ensembles is important for this branch of research to be able to more closely align with the needs of practitioners in the bioinformatics and network science communities.

Travelling salesman problem

This subsection, which is purely literature review, briefly explores the famous replica solution of the travelling salesman problem developed by Mézard and Parisi [1986]. Although the techniques are not taken further, they are included in order to present a more comprehensive literature review of statistical mechanics techniques which characterise and analyse the property of loops on a network. The problem is posed as: given N points and travel-costs (distances) between them d_{ij} distributed according to some distribution $\rho(d)$, what is the length of the shortest path that goes through all the points? The problem can be phrased as N points distributed on a unit square, but in any case assumes that as the number of nodes increases, the average

distance between them reduces. The authors make the assumption that the distribution of distances between points satisfies $\rho(d) \approx_{d\to 0} \frac{d^r}{r!}$, where r is a scaling parameter, from which the authors claim that the nearest neighbour of any point i is at a distance $d \approx N^{\frac{-1}{r+1}}$, and the length of the shortest tour will be of the order of $N^{1-\frac{1}{r+1}}$. The relevant partition function is of the form $Z_{TSP} = \sum_{tours} \exp\left(-\beta N^{\frac{-1}{r+1}} L_{tour}\right)$ where L_{tour} is the length of the tour or cycle. The strategy is that in the limit of the temperature $1/\beta$ going to zero, only the shortest tour will be picked out.

The next step requires an interesting strategy developed in polymer science by de Gennes [1972], and which has wider potential to allow for the analytical treatment of loops. We are invited to introduce a p-component spin on every site i of fixed length $S_i^2 = p$. It is now necessary to proceed in the same way as a high-temperature expansion for a vector spin system. If we write $U_{ij} = e^{-\beta^\star d_{ij}}$, then a high-temperature expansion following the exposition of Orland [1985] shows that the quantity to evaluate is

$$\zeta = \int \prod_{i=1}^{N} dS_i \prod_{\langle i,j\rangle} \left(1 + KU_{ij}\mathbf{S}_i \cdot \mathbf{S}_j\right) \tag{5.1.1}$$

where the $\mathbf{S_i}$ is a length p vector taking values $+/-1$. Orland [1985] characterises the expansion of this function as *"a factor of KU_{ij} per bond and a factor p per connected part"*, and this can be demonstrated by expanding the product in the normal way.

For the travelling salesman problem, we require configurations corresponding to a single 'polygon' going through every one of the N points. This is equivalent to seeking the part of the K^N coefficient which is proportional to p. This can be isolated by writing

$$Z_{TSP} = \lim_{K\to\infty} \lim_{p\to 0} \frac{1}{K^N} \frac{1}{p} \zeta \tag{5.1.2}$$

The first limit combined with the factor of $\frac{1}{K^N}$ truncates the series below K^N. The second limit combined with the factor $\frac{1}{p}$ removes any terms corresponding to disconnected polygons. In the $p = 0$ limit de Gennes [1972] finds that

$$\zeta = \int \prod_{i=1}^{N} \mathrm{d}S_i exp(K \sum_{\langle i,j\rangle} U_{ij} S_i S_j) \tag{5.1.3}$$

The de Gennes [1972] approach is the key insight that turns this calculation into a feasible problem, since it performs the function of efficiently book-keeping closed loops. From this point Mézard and Parisi [1986] can, with a few assumptions, bring in the replica machinery to be able to determine the non-vanishing contributions to the cost function. These papers, and

related works, provide a proof of concept of the power and potential of the replica approach in dealing with non-tree like topologies.

5.2 Strauss model

In this section we will look at the Strauss ensemble — the maximum entropy ensemble for a specified average degree and average number of triangles. We will briefly outline its key properties. We will then leverage the work of Burda et al. [2004b], and some simple approximations, to make some observations about the scaling behaviour and the critical points, culminating in new expressions for the entropy of this ensemble. The main drawback of the Strauss [1986] model, is that it has a condensed phase, where the network has a tendency to form complete cliques, which does not reflect the topology of real networks. Burda et al. [2004b] refined the understanding of this phenomena, showing that the clustered phase occurred above certain critical values of the parameters. We briefly discuss how the Strauss model could be extended, and propose that such a generalisation could be approached via a novel extension of the replica technique. We demonstrate that the Strauss model is both an effective and an analytically tractable way of introducing loops into a random graph ensemble, but real networks (e.g. protein-protein interaction networks) typically have many more loops than the sub-critical parameters can achieve. The ensemble is defined via

$$p(\boldsymbol{c}) = Z^{-1}(u,g)\mathrm{e}^{uTr(\boldsymbol{c}^2)+gTr(\boldsymbol{c}^3)} \tag{5.2.1}$$

$$Z(u,g) = \sum_{\boldsymbol{c}} \mathrm{e}^{uTr(\boldsymbol{c}^2)+gTr(\boldsymbol{c}^3)} \qquad \phi(u,g) = \frac{1}{N}\ln Z(u,g) \tag{5.2.2}$$

for parameters u and g to be selected in order to tune the ensemble averages by using the relationships, and noting that by the symmetry of $\boldsymbol{c}$ we know that $Tr(\boldsymbol{c}^2) = \sum_{ij} c_{ij}c_{ji} = \sum_{ij} c_{ij}$

$$\langle k\rangle = \frac{\partial}{\partial u}\phi(u,g) \quad \langle m\rangle = \frac{\partial}{\partial g}\phi(u,g) \tag{5.2.3}$$

For this ensemble, we wish to calculate the Shannon entropy $S = -\sum_{\boldsymbol{c}} p(\boldsymbol{c})\log p(\boldsymbol{c})$ in leading order in N. Applied to the above expression this immediately yields

$$S = \left[1 - u\frac{\partial}{\partial u} - g\frac{\partial}{\partial g}\right]\log Z(u,g) = N\left[\phi(u,g) - u\langle k\rangle - g\langle m\rangle\right] \tag{5.2.4}$$

We will now concentrate on studying the free energy $\phi(u, g)$, since all the other important parameters follow directly. Setting g to be equal to zero retrieves the Erdös-Rényi ensemble

$$p_{ER}(\boldsymbol{c}) = Z^{-1}(u)\mathrm{e}^{uTr(\boldsymbol{c}^2)} \qquad Z_{ER}(u) = \sum_{\boldsymbol{c}} \mathrm{e}^{uTr(\boldsymbol{c}^2)} \tag{5.2.5}$$

$$\ln Z_{ER} = \ln \sum_{\boldsymbol{c}} \mathrm{e}^{u\sum_{ij} c_{ij}} = \frac{N(N-1)}{2} \ln\left(\mathrm{e}^{2u} + 1\right)$$

Differentiating and substituting in $u = -\frac{1}{2}\ln\left(\frac{1-p}{p}\right)$ it immediately follows that

$$\frac{\partial}{\partial u}[\ln Z_{ER}] = \frac{N(N-1)\mathrm{e}^{2u}}{\mathrm{e}^{2u}+1} = N(N-1)p \tag{5.2.6}$$

This justifies the choice of substitution, since it is now possible to recognise that the parameter p can be interpreted as the likelihood of a link between two nodes in the Erdös-Rényi ensemble, and is related to the Erdös-Rényi average degree via $p = \bar{k}/N$. This can be used to motivate rewriting equation 5.2.2 as

$$\phi(u, g) = \frac{1}{N}\ln\textstyle\sum_{\boldsymbol{c}} p_{ER}(\boldsymbol{c})\mathrm{e}^{gTr(\boldsymbol{c}^3)} + \frac{1}{N}\ln Z_{ER}(u) = \frac{1}{N}\ln\langle \mathrm{e}^{gTr(\boldsymbol{c}^3)}\rangle_{ER} + \frac{1}{N}\ln Z_{ER}(u) \tag{5.2.7}$$

hence it is evident that the substance of the problem is to determine the moments of the distribution of triangle counts in the Erdös-Rényi ensemble.

$$\phi(u, g) = -\frac{1}{2}(N-1)\log\left(\mathrm{e}^{2u}+1\right) + \frac{1}{N}\log\sum_{r\geq 0} p(r|u)\mathrm{e}^{gr} \tag{5.2.8}$$

with $p(r|u) = \langle \delta_{r,\mathrm{Tr}(\boldsymbol{c}^3)}\rangle_{\bar{k}}$ as the distribution of the random variable $\mathrm{Tr}(\boldsymbol{c}^3) \in \mathbb{N}$ for Erdös-Rényi graphs with average degree $\bar{k} = N/(\mathrm{e}^{-2u}+1)$. The conditional average is calculated easily

$$\bar{r}(u) = \sum_{r\geq 0} r p(r|u) = \Big\langle \sum_{ijk} c_{ij}c_{jk}c_{ki}\Big\rangle = \bar{k}^3 + O(N^{-1}) \tag{5.2.9}$$

As a first approximation, we will complete this calculation under the ansatz that the average triangle count over the Erdös-Rényi ensemble is distributed according to a Poissonian distribution. If we approximate $p(r|u)$ by $p(r|u) \approx \mathrm{e}^{-\bar{r}(u)}[\bar{r}(u)]^r/r!$, we can do the sum over r in equation 5.2.8

$$\phi(u, g) \approx \frac{1}{N}(\mathrm{e}^g - 1)\bar{r}(u) + \frac{1}{N}\ln Z_{ER} \tag{5.2.10}$$

Using the relations from equation 5.2.3, we can immediately observe that

$$\langle k\rangle \approx \bar{k}\left(1 + \frac{6\mathrm{e}^g\bar{k}^3}{N} + O\left(N^{-1}\right)\right) \tag{5.2.11}$$

$$\bar{m} = \langle m\rangle \approx \frac{\mathrm{e}^g\bar{k}^3}{N}\left(1 + O\left(N^{-1}\right)\right)$$

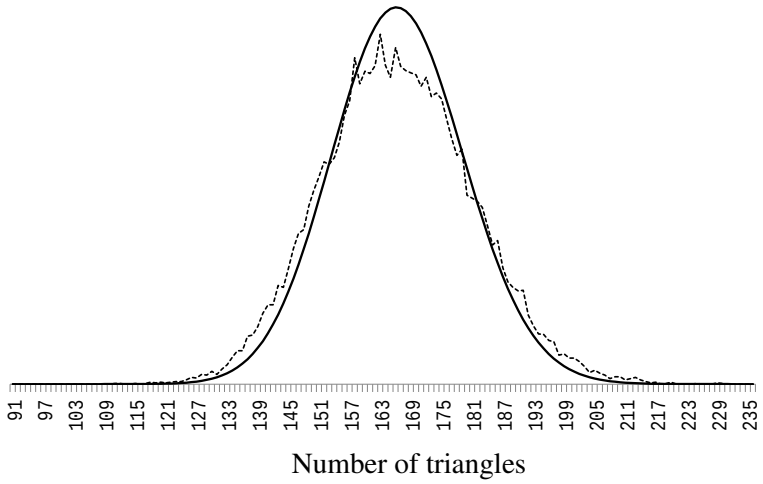


Figure 5.2: Distribution of triangle counts based on sampling 20,000 networks from an Erdös-Rényi ensemble ($N = 1,000$, $\bar{k} = 10$). A Poissonian distribution with the correct average number of triangles is plotted over the top of this. It can be seen by inspection that the real distribution is flatter, with outlier values more likely than the Poissonian distribution would suggest.

for $\bar{k}$ being the average connectivity of the Erdös-Rényi ensemble associated with u. For the model to achieve $O(N)$ number of triangles, g must scale like $\log N$. Equivalently,

$$u = \frac{1}{2}\ln(\langle k\rangle - 6\bar{m}) - \frac{1}{2}\ln N + O(N^{-1}) \tag{5.2.12}$$

The corresponding leading order approximation for the entropy per node is

$$S_N \approx \left(\frac{1}{2}\langle k\rangle - \bar{m}\right)\log N - \frac{1}{2}(\langle k\rangle - 6\bar{m})\log(\langle k\rangle - 6\bar{m}) - \bar{m}\log\bar{m} + \frac{1}{2}\langle k\rangle - 2\bar{m} \tag{5.2.13}$$

This is a first estimate, and a plot of the actual average triangle distribution (figure 5.2) within a sample Erdös-Rényi ensemble does not provide strong support to the assumption of this being a Poissonian distribution.

The authors of Burda et al. [2004b] proposed a diagrammatic approach to evaluate the expected values of the moments of the triangle distribution. They evaluated the free energy as

$$\ln Z(G,\gamma) = \gamma(\mathrm{e}^G - 1) + \ln Z_{ER} \tag{5.2.14}$$

where we can identify $\gamma = \frac{\bar{k}^3}{6}$ for $\bar{k} = pN$, and $G = 6g$. The average value of $Tr(\boldsymbol{c}^3)$ in this

network can be evaluated as

$$\frac{\partial}{\partial g}\ln Z_G = \bar{k}^3 e^G = \langle m\rangle N = 6\langle T\rangle \tag{5.2.15}$$

with the substitution $\langle T\rangle = \frac{\langle m\rangle N}{6}$ reflecting scaling. We know from Burda et al. [2004b] that if we make the substitution $G^\star \log N + \alpha = G$ for $G^\star$ and α being functions of the degree without N dependency, then the perturbation series will break down when $G^\star$ is strictly less than 1 (numerically found to be about 0.7). Hence, we can see that within this range $N^{G^\star - 1} \to 0 \quad \text{as} \quad N \to \infty$. This means that the number of triangles per node tends to zero in the large N limit. To derive the change in the average degree compared to the associated Erdös-Rényi ensemble, we calculate

$$\langle k\rangle - \bar{k} = \frac{\partial}{\partial u}\left[\frac{1}{N}\ln Z_G\right] = \bar{k}^3(1-p)\frac{(e^G-1)}{N} \approx \frac{6\langle T\rangle}{N} \text{ for N} \to \infty \tag{5.2.16}$$

Hence, in the leading order for large N the average degree is unchanged, and the average number of triangles per node is zero. However, they both disappear at the same rate. Figure 7 of Burda et al. [2004b] presented evidence that the degree distribution is insensitive to the values of the coupling parameter. There is no contradiction with the result above - it is merely a consequence of the fact that sub-critical parameters achieve only very low values of average triangles per node. The remaining expression to evaluate is, with suitable change of variables,

$$C_G \quad = \quad S_{full} - S_{ER} = \left[1 + p(1-p)\ln\left(\frac{1}{p}-1\right)\frac{\partial}{\partial p} - G\frac{\partial}{\partial G}\right]\frac{\bar{k}^3}{6}(e^G-1) \tag{5.2.17}$$

which equates to

$$C_G \quad = \quad \frac{\bar{k}^3}{6}(e^G-1) + \frac{1}{2}\ln\left(\frac{1}{p}-1\right)\bar{k}^3(1-p)(e^G-1) - \frac{G}{6}\bar{k}^3 e^G \tag{5.2.18}$$

Returning back to calculating $S_{full} = C_G + S_{ER}$ taking the leading order expansion of the logarithm, and eliminating the parameter G with equation 5.2.15 it follows that

$$S = \frac{\bar{k}(N-1)}{2}\left[1+\ln\left(\frac{N}{\bar{k}}\right)\right] + \frac{\langle T\rangle}{6}\left[1+\ln\left(\frac{N^3}{\langle T\rangle}\right)\right] - \frac{\bar{k}^3}{6}\left[1+\ln\left(\frac{N^3}{\bar{k}^3}\right)\right] - \frac{3\bar{k}^2}{4} + O(N^{-\epsilon}) \tag{5.2.19}$$

This form has pleasing symmetries and similarities with previously derived results (e.g. chapter 3). If $\langle T\rangle = \bar{k}^3$ then the entropy reduces down to the Erdös-Rényi entropy, as expected. Direct comparisons with equation 5.2.13 are not valid, because the expressions refer to different scalings of $\langle m\rangle$. The next step would be to eliminate $\bar{k}$ in favour of the observable $\langle k\rangle$. This is not so simple - as it effectively requires the solution of a fourth order equation. Taking a leading order approximation results in an error that is unacceptably large compared to the overall answer.

The authors of Burda et al. [2004b] numerically deduced the critical values for the coupling parameter G for networks with an average degree of 2, 4 and 8. This gives us a region within which we know that it is valid to apply the formula 5.2.19 - and indeed reasonable to use a model with such parameters to model a real network. We constructed a spreadsheet to evaluate equation 5.2.19 and related quantities for values of the parameter G up to the critical point. The implied parameter $\bar{k}$ is deduced numerically. The results are charted in figures 5.3 and 5.4. These charts show that, for these (reasonable) values of average degree and network size, by the time the coupling constant reaches the critical value, the number of triangles in the network is predicted to increase by more than tenfold compared to an Erdös-Rényi ensemble with the same average degree. Nonetheless - the complexity is low when viewed as a proportion of the overall entropy of the ensemble. This reflects that constraining only the average number of loops, and remaining within a phase with relatively low clustering, still leaves a lot of topological freedom for the networks in the ensemble. In order to form a view on how applicable the Strauss model

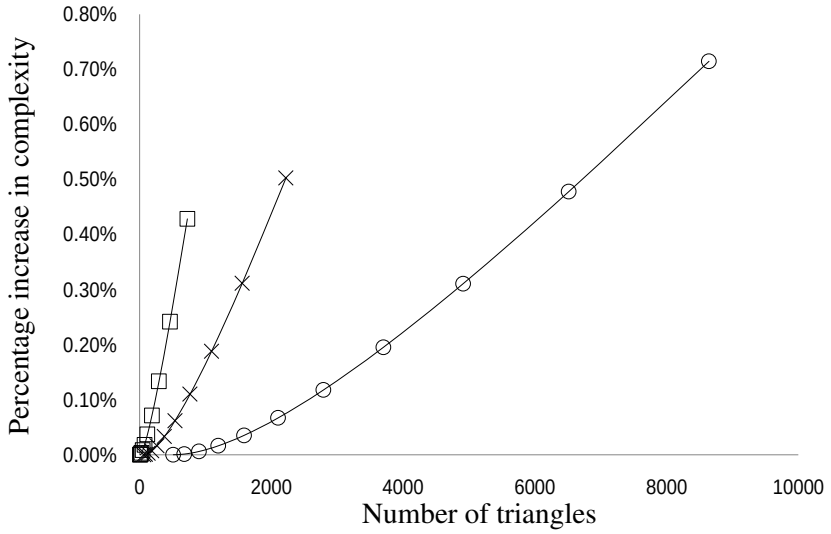


Figure 5.3: Complexity and predicted number of triangles for values of the coupling constant G below the transition point to the clustered phase. From the left, the lines correspond to $\bar{k} = 2$ (squares), $\bar{k} = 4$ (crosses) and $\bar{k} = 8$ (circles). $N = 10,000$ in each case. The x-axis plots $\langle Tr(\boldsymbol{c}^3)\rangle$. The y-axis plots the ratio of the complexity associated with triangle action term, and the entropy of the Erdös-Renyi ensemble with the same average degree and network size.

is to modelling biological networks, figure 5.5 compares the number of triangles for biological

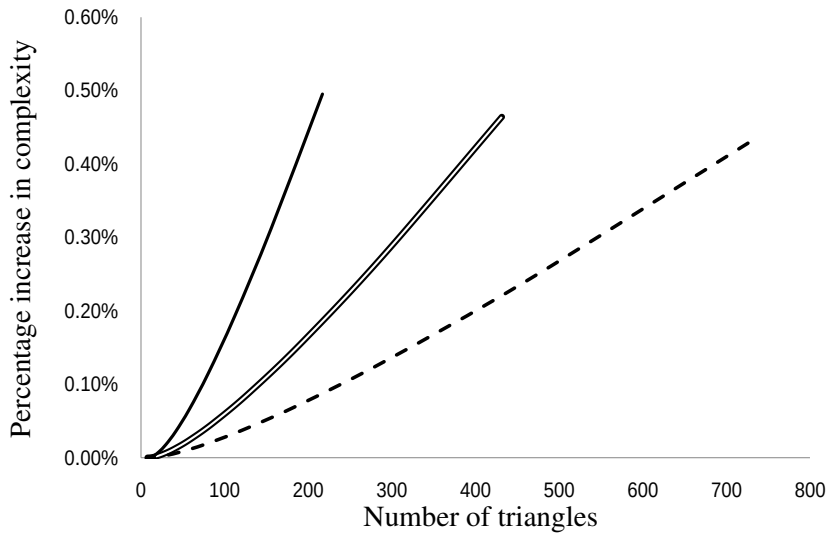


Figure 5.4: Complexity and predicted number of triangles for values of the coupling constant G below the transition point to the clustered phase. From the left, the lines correspond to $N = 2,000$ (single line), $N = 5,000$ (double line) and $N = 10,000$ (dashed line). In all cases $\bar{k} = 2$. The x-axis plots $\langle Tr(\boldsymbol{c}^3)\rangle$. The y-axis plots the ratio of the complexity associated with triangle action term, and the entropy of the Erdös-Renyi ensemble with the same average degree and network size.

networks whose average degree lies between 2 and 8, with the maximum number of triangle that a Strauss-type model could be expected to generate, before it goes into its degenerate clustered phase. This table suggests that, while the Strauss model is a substantial improvement on the Erdös-Rényi model from the point of view of the number of loops, it unfortunately typically collapses into its clustered phase before it reaches biologically realistic values for these parameters.

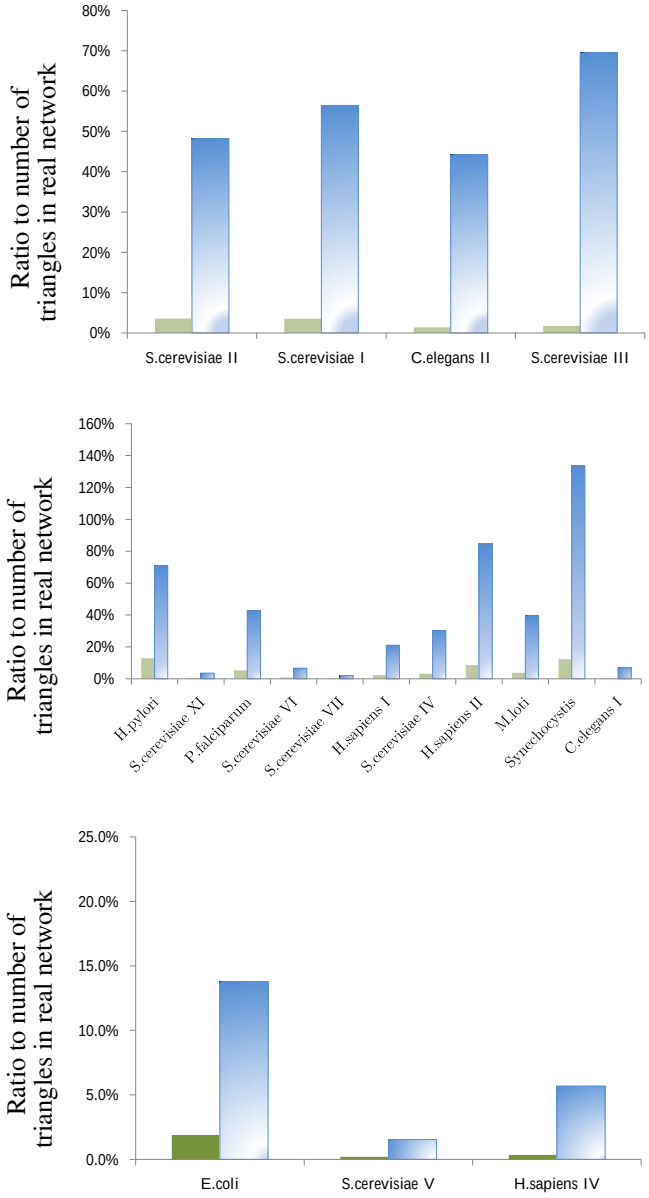


Figure 5.5: The solid bar gives the percentage value of (number of triangles in the Erdös-Renyi ensemble with the same average degree)/ (actual number of triangles). The graduated bar gives the percentage value of (estimated maximum number of triangles that the Strauss model can generate before clumping sets in)/(actual number of triangles). From the top, the charts show networks which have average degree roughly 2, 4 and 8 respectively. On each chart, the columns are ordered from left to right in terms of increasing network size (average network size: 2,000 nodes).[a]

[a]Nomenclature and references for the datasets is the same as Fernandes et al. [2010], which also has broader discussion of the topological properties of these particular networks. Triangle counts kindly provided by Sun Sook Chung from the Fraternali Group.

If we wished to extend the perturbation analysis beyond the critical point, we would need to look at different scalings of the parameter g. However, since the un-physical behaviour of the Strauss model above this point has already been shown, such a result would be of limited application. The authors of Burda et al. [2004b] themselves extended their analysis in Burda et al. [2004a] to consider general uncorrelated degree distributions - but the agreement with simulations was less precise. A general degree distribution would make the model more flexible, but would not necessarily increase the number of triangles that the model could generate.

Figure 5.6 gives an intuitive illustration why the Strauss model clumps above a certain critical point. Based on this logic, one way to moving the critical point higher, and designing a more versatile model, may be to introduce extra terms into the Hamiltonian which refer to higher order motifs. The early steps of the argument in Burda et al. [2004b] apply: it would be possible to define a diagramatic expansion and to identify the leading order diagrams. Unfortunately, it is then not clear how to analytically re-sum the contributing terms. A more general form of this would be to write

$$p(\boldsymbol{c}) = Z^{-1}[\hat{\rho}]\mathrm{e}^{N\int \mathrm{d}\mu \hat{\rho}(\mu)\rho(\mu|\boldsymbol{c})} \quad \text{for} \quad \rho(\mu|\boldsymbol{c}) = \frac{1}{N}\sum_i \delta\left[\mu - \mu_i(\boldsymbol{c})\right] \tag{5.2.20}$$

where $\rho(\mu|\boldsymbol{c})$ is the spectrum of $\boldsymbol{c}$, and the function $\hat{\rho}(\mu)$ drives the properties of the model. For the Strauss case, $\hat{\rho}(\mu) = u\mu^2 + v\mu^3$. Including higher order terms (e.g. coefficient of μ^4 to influence the average number of closed paths length 4, and so on) would give better control over the clumping of the Strauss model. Determining the entropy of this model would require the evaluation of

$$\phi_N = \frac{1}{N}\log\sum_{\boldsymbol{c}} \mathrm{e}^{N\int \mathrm{d}\mu\ \hat{\varrho}(\mu)\varrho(\mu|\boldsymbol{c})} \qquad \hat{\varrho}(\mu) = u\mu^2 + \sum_{\ell=3}^{L} v_\ell \mu^\ell \tag{5.2.21}$$

Calculating the sum over all graphs $\boldsymbol{c}$ in equation 5.2.21 directly is not feasible, since the argument of the sum does not factorise over the link variables. However, we can rewrite ϕ_N using the standard spectrum formulae from Edwards and Jones [1976]

$$\varrho(\mu|\boldsymbol{c}) = \frac{2}{N\pi}\lim_{\varepsilon\downarrow 0}\mathrm{Im}\frac{\partial}{\partial\mu}\log Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c}) \qquad Z(\mu|\boldsymbol{c}) = \int \mathrm{d}\boldsymbol{\phi}\ \mathrm{e}^{-\frac{1}{2}\mathrm{i}\boldsymbol{\phi}\cdot[\boldsymbol{c}-\mu\,\mathbb{I}]\boldsymbol{\phi}} \tag{5.2.22}$$

which gives us, after integration by parts in the exponent,

$$\phi_N \quad = \quad \lim_{\varepsilon\downarrow 0}\frac{1}{N}\log\sum_{\boldsymbol{c}} \mathrm{e}^{u\sum_{ij}c_{ij} - \frac{2}{\pi}\sum_{\ell=3}^{L}\ell v_\ell \int \mathrm{d}\mu\ \mu^{\ell-1}\mathrm{Im}\log Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})} \tag{5.2.23}$$

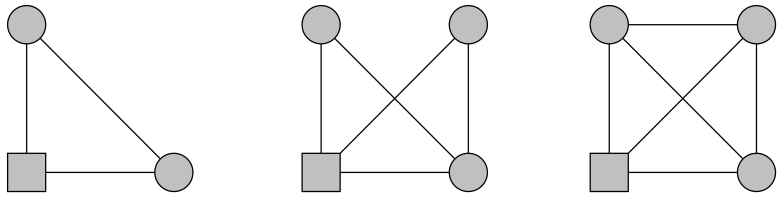


Figure 5.6: This diagram is provided to support intuition on why the Strauss model clumps when the parameters are tuned to induce average number of triangles greater than one triangle per node. From the left, the first diagram has three edges and one triangle involving the square node. The next diagram has two triangles, at a 'cost' of two extra edges. The final diagram has one additional edge - six in total - arranged to form four triangles. In this way, the system will favour forming triangles in regions which are already triangle-dense, ultimately leading to a large fully connected clique, rather than a more even triangle distribution throughout the network. This may be able to be controlled by including some higher order constraints to inhibit the formation of cliques.

We discretize the integral via $\int \mathrm{d}\mu \to \Delta \sum_\mu$, and use the identity $\mathrm{e}^{-2\mathrm{Im}\log z} = z^{\mathrm{i}}/\overline{z}^{\mathrm{i}}$:

$$\begin{aligned}
\phi_N &= \lim_{\varepsilon,\Delta\downarrow 0} \frac{1}{N} \log \sum_{\boldsymbol{c}} \mathrm{e}^{u\sum_{ij} c_{ij}} \prod_\mu \Big[\mathrm{e}^{-2\mathrm{Im}\log Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})}\Big]^{\frac{\Delta}{\pi}\sum_{\ell=3}^{L}\ell v_\ell \mu^{\ell-1}} \\
&= \lim_{\varepsilon,\Delta\downarrow 0} \frac{1}{N} \log \sum_{\boldsymbol{c}} \mathrm{e}^{u\sum_{ij} c_{ij}} \prod_\mu \Big[Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})^{\mathrm{i}}\,\overline{Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})}^{-\mathrm{i}}\Big]^{\frac{\Delta}{\pi}\sum_{\ell=3}^{L}\ell v_\ell \mu^{\ell-1}} \\
&= \lim_{\varepsilon,\Delta\downarrow 0} \lim_{n_\mu\to\frac{\Delta\mathrm{i}}{\pi}\sum_{\ell=3}^{L}\ell v_\ell\mu^{\ell-1}} \lim_{m_\mu\to -n_\mu} \Phi_N(\{n_\mu,m_\mu\})
\end{aligned} \tag{5.2.24}$$

in which

$$\Phi_N(\{n_\mu,m_\mu\}) = \frac{1}{N}\log\sum_{\boldsymbol{c}} \mathrm{e}^{u\sum_{ij}c_{ij}} \prod_\mu \Big[Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})^{n_\mu}\,\overline{Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})}^{m_\mu}\Big] \tag{5.2.25}$$

Expression 5.2.25 is reminiscent of formulae encountered in replica analyses of heterogeneous many-variable systems (see e.g. Mézard et al. [1987]), which suggests a strategy for proceeding with the calculation. If we could carry out the sum over all graphs $\boldsymbol{c}$ by evaluating equation 5.2.25 for positive integer values of $\{n_\mu, m_\mu\}$ (where the powers of $Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})$ and $\overline{Z(\mu+\mathrm{i}\varepsilon|\boldsymbol{c})}$ become multiple *replicated* Gaussian integrals), then the full expression could be determined by taking the limits required in 5.2.24 by analytical continuation.

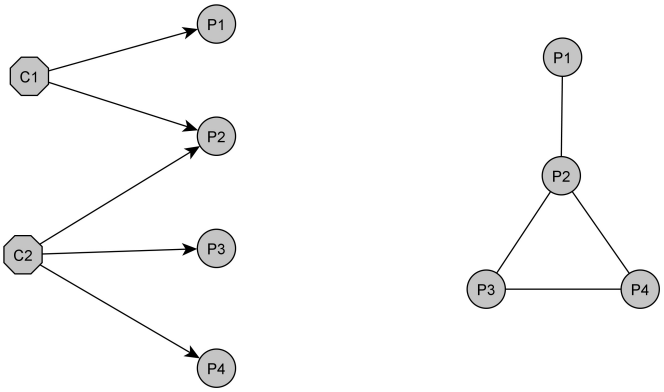


Figure 5.7: A bipartite protein complex model, and its projection to the protein-protein interaction space. On the left, the bipartite graph's left hand column is identified as protein complexes, and the right hand column is proteins. There is a link if the particular protein is a constituent of the protein complex. The right hand network shows the projection of the bipartite network into protein space. There is a link between two proteins if they are part of the same protein complex. The model can be formulated with or without double links being allowed.

5.3 Protein complex model

This section is based on the model developed by Agliari et al. [2013a,b] in the context of immune models. Newman [2003] independently looked at a similar model, but from the point of view of studying the expected topological properties of such networks (e.g. size of giant components, epidemic threshold). A major class of real-world networks to which we would hope to apply techniques to is protein-protein interaction networks (PPIN). With this end-goal in mind, we will re-formulate the model of Agliari et al. [2013a,b] as a bipartite network of protein complexes, where the PPIN is the projection onto the protein space. In this way, although we lose the clarity of working with the Strauss maximum entropy ensemble, we gain a richly intuitive framework to model real-life phenomena.

We define a bipartite graph, mapping from αN protein complexes, to their N constituent proteins. The variable ξ_i^μ can take values of $\{1, 0\}$ - indicating whether the protein complex indexed μ contains the protein indexed i. We can define an induced network $\boldsymbol{c}$ on the protein nodes by specifying that $c_{ij} = \sum_{\mu=1}^{\alpha N} \xi_i^\mu \xi_j^\mu$. It is known from Agliari et al. [2013a] that the induced network on the proteins has a high degree of clustering, and hence a non-trivial number of short-loops. However, the underlying bipartite model is tree-like, and thus well behaved under normal ana-

lytical techniques. Hence, this provides a promising route to a solvable model which captures the clustering properties of real PPIN.

Define the probability of observing a network $\boldsymbol{c}$.

$$p(c) = \left\langle \prod_{i<j} \delta_{c_{ij},\sum \xi_i^\mu \xi_j^\mu} \right\rangle_{\{\xi\}} \tag{5.3.1}$$

where the ξ_i^μ are drawn independently from

$$p(\xi) = \frac{q}{N}\delta_{\xi,1} + \left(1 - \frac{q}{N}\right)\delta_{\xi,0} \tag{5.3.2}$$

Several of the observables on the projected network $\boldsymbol{c}$ can be immediately deduced from the parameters specified for the bijective graph.

$$\left\langle c_{ij} \right\rangle = \alpha N \left\langle \xi_i \xi_j \right\rangle = \frac{\alpha q^2}{N} \tag{5.3.3}$$

$$p(c_{ij}) = \int_{-\pi}^{\pi} \frac{\mathrm{d}\omega}{2\pi} \mathrm{e}^{\mathrm{i}\omega c_{ij}} \left\langle \mathrm{e}^{\mathrm{i}\omega \xi_i \xi_j} \right\rangle_{\{\xi\}}^{\alpha N} = \sum_{l=0}^{\alpha N} \delta_{c_{ij},l} \binom{\alpha N}{l} \left[\frac{q^2}{N^2}\right]^l \left[1 - \frac{q^2}{N^2}\right]^{\alpha N - l} \tag{5.3.4}$$

from which it follows that $\bar{k} = N^{-1} \sum_{ij} \left(1 - p(c_{ij} = 0)\right) = \alpha q^2$. The benefit of this model is that you get loops in the projection in protein space from a model which is tree-like when viewed in the protein-complex space. Figure 5.8 uses the data on protein complexes found in yeast in order to construct a protein-protein interaction network in the way described in this section. We will now approach evaluating the Shannon Entropy of the ensemble of networks projected in the protein space.

$$S = -\sum_{\boldsymbol{c}} \left[\sum_{\boldsymbol{\xi}} p(\boldsymbol{\xi}) \prod_{i<j} \delta\left(\sum_{\mu} \xi_i^\mu \xi_j^\mu - c_{ij} \right) \right] \log \left[\sum_{\boldsymbol{\xi}} p(\boldsymbol{\xi}) \prod_{i<j} \delta\left(\sum_{\mu} \xi_i^\mu \xi_j^\mu - c_{ij} \right) \right] \tag{5.3.5}$$

Although Agliari et al. [2013a] evaluated the free energy of the bipartite model, this cannot be directly applied to determine the entropy of the ensemble of networks projected onto the protein space because more than one network in the bipartite space corresponds to every network in the protein space. In the next two subsections we set out some ideas for a formal and a heuristic approach respectively.

5.3.1 Replica approach

The replica method is based on the identity $\langle \log Z \rangle = \lim_{n\to 0} \frac{1}{n} \log \langle Z^n \rangle$ (see Mézard et al. [1987] and citing papers for justification and some previous applications). The Shannon entropy can

be trivially re-written as an average across the ensemble.

$$S = \frac{1}{N}\sum_{\boldsymbol{c}} p(\boldsymbol{c})\log p(\boldsymbol{c}) = \frac{1}{N}\langle \log p(\boldsymbol{c})\rangle_{\boldsymbol{c}} \tag{5.3.6}$$

This form allows us to apply the replica identity

$$S = \lim_{n\to 0}\frac{1}{nN}\log\langle p^n(\boldsymbol{c})\rangle_{\boldsymbol{c}} \tag{5.3.7}$$

The expression inside the logarithm can be expanded

$$\langle p^n(\boldsymbol{c})\rangle_{\boldsymbol{c}} = \sum_{\boldsymbol{c}}\sum_{\{\xi_1\}}\cdots\sum_{\{\xi_{n+1}\}}\prod_{s=1}^{n+1} p(\{\xi_s\})\prod_x\prod_{ij}\delta_{c_{ij},\sum_\mu {}^x\xi_i^\mu {}^x\xi_j^\mu} \tag{5.3.8}$$

If we were to immediately re-sum from this point, then this would result in a series solution, very similar in substance to the heuristic approach outlined in the next section. However, based on previous similar calculations, we would expect to be able to simplify substantially if we were to introduce a carefully chosen measure which allowed us to benefit from the finite connectivity of the ensemble, in order to be able to efficiently and rigorously discard non-leading order terms.

5.3.2 Clique decomposition to estimate a network's statistical weight

Below we propose a rough heuristic approach to this problem by observing that the most likely bipartite network which leads to $\boldsymbol{c}$ can be identified as the bipartite network associated with the minimal clique covering of $\boldsymbol{c}$. If we add strong assumptions about the network being dilute, and links within the bipartite network being uncorrelated, we can estimate the Shannon entropy combinatorially.

A *clique cover* of G is a set of cliques that cover the edge set of G. It is clear that every projection from the bipartite protein complex space to the protein space corresponds to a clique cover, where every clique is associated with a protein complex μ. The total number of nodes in the clique cover determines the statistical weight of this configuration. Define a *minimal clique cover* to be the the clique cover which requires the fewest cliques. We claim without proof that every network has a unique minimal clique cover, up to permutation of multiple edges. Every minimal clique cover can be broken down into a covering which involves a larger number of smaller cliques. This will precisely enumerate the bipartite networks which can induce a particular network $\boldsymbol{c}$. The major benefit of this approach is that we can remove the summation from inside the logarithm in equation 5.3.5. Instead of summing over every combination of

$\boldsymbol{\xi}$, we know that the only contributing elements will come from the cliques constituting the minimal clique cover $Q(\boldsymbol{c})$, and sub-cliques thereof. So we can rewrite the expression above as

$$S = -\sum_{\boldsymbol{c}} F[Q(\boldsymbol{c})] \log F[Q(\boldsymbol{c})] \tag{5.3.9}$$

$$F[Q(\boldsymbol{c})] = \binom{\alpha N}{s} \left(\frac{q}{N}\right)^{Q^\star} \left(1-\frac{q}{N}\right)^{O(\alpha N^2)} + O(N^{s-Q^\star-1}) \tag{5.3.10}$$

Where $Q^\star$ is the sum of the sizes of the cliques constituting the minimal clique cover $Q(\boldsymbol{c})$ and s is the number of such cliques. The difficulty that is hidden in this form is that we do not know how many minimal coverings there are of a particular size. However, given a network $\boldsymbol{c}$ we can identify its minimal clique covering, so it is meaningful to write

$$S = \mathrm{e}^{-\alpha q N} \sum_{\boldsymbol{c}} \sum_{Q=2}^{\infty} \sum_{s>0} \delta_{Q,Q^\star(\boldsymbol{c})} \delta_{s,s(Q(\boldsymbol{c}))} \binom{\alpha N}{s} \left(\frac{q}{N}\right)^{Q} [(Q-s)\log N + \alpha N q + O(1)] - p_0 \log p_0 \tag{5.3.11}$$

where p_0 is a place-holder for the probability of $\boldsymbol{c}$ being an empty network. The $\boldsymbol{c}$ dependence is now isolated within

$$\sum_{\boldsymbol{c}} \delta_{Q,Q^\star(\boldsymbol{c})} \delta_{s,s(Q(\boldsymbol{c}))} \approx \frac{N^Q}{Q!} \sum \frac{Q!}{Q_1! \ldots Q_s!} \tag{5.3.12}$$

the number of networks that have a minimal clique decomposition made up of s cliques of combined size $Q^\star$, which can be estimated as the number of ways of pick Q nodes (multiple occupancy allowed) multiplied by the number of ways of partitioning Q nodes across s sets.

Since the purpose at this time is to just explore the idea of approaching this calculation via clique decomposition, we can satisfy ourselves with observing that the biggest term in the s summation for any given Q corresponds to the largest possible size of s: $\lfloor \frac{Q}{2} \rfloor$ corresponding to dividing the Q nodes across cliques of size 2. In fact, the preceding steps can be roughly summarised as observing that the biggest cohort from the ensemble will consist of networks with dimers (disjoint links), the next biggest term will arise from networks that have dimers and trimers (links and triangles), and so on. We will continue the calculation based on just this largest term, being aware that there will remain residual corrections to the leading order.

$$S \approx \mathrm{e}^{-\alpha q N} \sum_{Q=2}^{\infty} \binom{\alpha N}{Q^\star/2} \left[\frac{q^2}{2}\right]^{Q^\star/2} \left[\frac{q}{3}\right]^{Q-Q^\star} \left[(Q-Q^\star/2)\log N + \alpha N q\right] \tag{5.3.13}$$

By inspection, we can see that this is a binomial type series. It is straightforward to sum the components relating to odd and even Q and assemble all the constituent terms to arrive at our

first estimate of the first leading order of the Shannon entropy of this ensemble.

$$S \approx \left(1 + \frac{q^2}{2}\right)^{\alpha N} \mathrm{e}^{-\alpha q N} \left(\left(\alpha N q + \frac{q^2}{2} \log N\right)\left(1 + \frac{q}{3}\right) + \frac{q}{3} \log N\right) - p_0 \log p_0 \tag{5.3.14}$$

This is a basic estimate based on a heuristic argument, which has made strong assumptions about independence and dilution, and only calculated the largest terms contributing to the series. The evaluation of equation 5.3.12 could be improved either by a direct calculation (e.g. such as set out in section 5.3.1), or by finding an efficient way of enumerating and summing all the possible minimal clique decompositions.

5.3.3 Next steps for the protein complex model

In Newman [2003], a similar model was extended so that in the projection of the network into the protein-protein space, a link occurred between nodes i and j with a probability driven by $\sum \xi_i^\mu \xi_j^\mu$. This allows for incomplete sampling and/or not every protein co-occuring in a complex having positive binary interaction. This generalisation would make the model outlined in section 5.3 more realistic. Sampling is always a consideration when working with real-world networks. The general approach to this (e.g. Annibale and Coolen [2011]) is to consider that for every c_{ij} there may be some possibility of a false positive (i.e. c_{ij} wrongly reported as 1) or a false negative. The protein complex network formulation allows the sampling question to be posed in a different way: as the possibility of having failed to identify (or wrongly identify) certain complexes. For example, if larger complexes were known to degrade before they could be analysed, or if binary transient interactions were under- or over- identified. This would be a meaningful formulation particularly for mass spectroscopy datasets. Figure 5.9 suggests that the assumption that links in the bipartite network occur at random is too strong, since this would imply a Poissonian distribution for complex sizes.

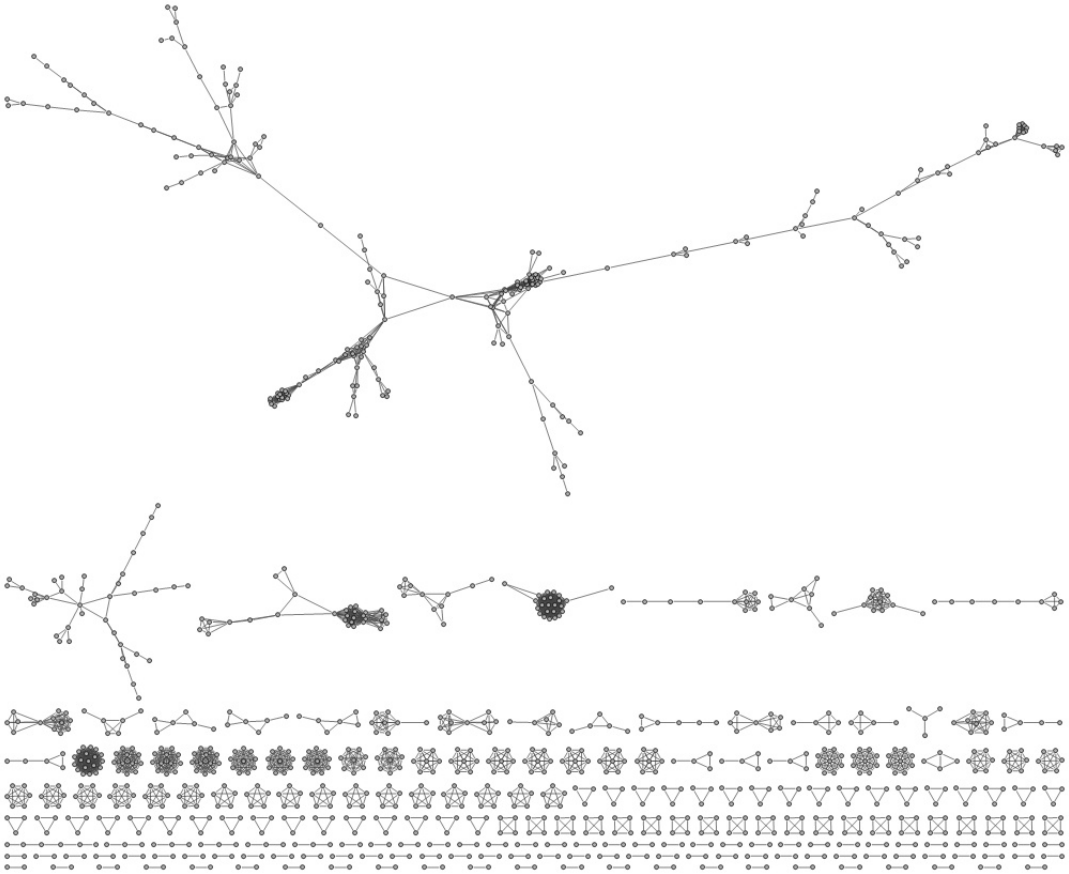


Figure 5.8: A projection onto the protein space of the protein complexes and their core constituents identified in *Sac. cerevisiae* by Gavin et al. [2006] via mass-spectroscopy. The network shows distinct cliques, but only the beginning of the dense core that is generally seen in protein-protein interaction networks. The dense core would be likely to emerge if the data included non-core proteins, and reactions which may not necessarily correspond to named complexes.

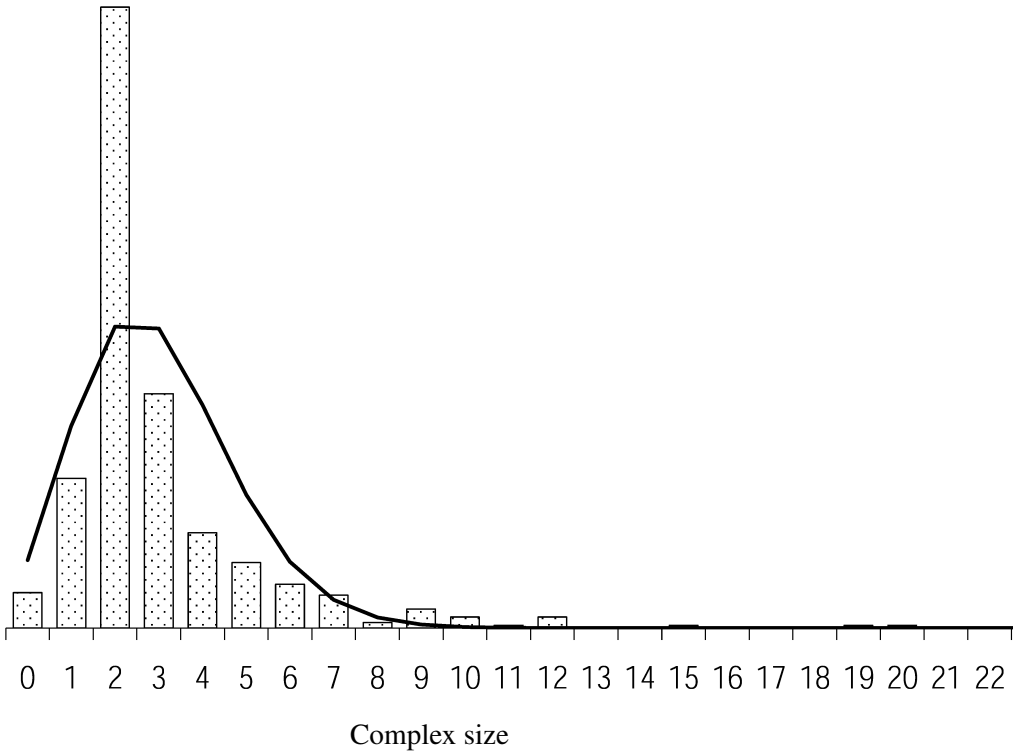


Figure 5.9: Distribution of complex sizes for the protein complex dataset for *Sac. cerevisiae* from Gavin et al. [2006], and as visualised in figure 5.8. The data was taken to include only core proteins, appearing in at least 2/3rds of all the isoforms (hence a zero complex size implies that there is no protein that is found in at least 2/3rds of all the isoforms). This shows a nascent power law degree distribution. Hence the theoretical model could be improved if the bipartite degree distribution could be specified. A Poissonian line graph is overlaid for comparison.

This model lends itself to physical interpretations of the parameters. It would be equally natural to use this framework to model a citation network, where the left hand side of the bipartite network represents publications, and the right hand side of the bipartite network represents authors. Projection onto author space give the well known collaboration network (e.g. Newman [2001]). Questions about contagion spreading in a network could be studied by first considering membership of social groups (e.g. families, workplaces, clubs). Two individuals could be considered to be linked, in the sense of risk of contagion spread, if they are members of the same social group. In both of these cases, estimates and models of the parameters of the bipartite network (e.g. average number of authors per paper, average number of people in a club) may be easier to obtain than estimates of the topological parameters on the projected network (e.g. average number of lifetime collaborators, average number of social contacts). The analytical tractability and intuitive structure of this model makes it a strong candidate to be implemented into real life applications.

5.4 Summary

In this chapter we have presented the Strauss model and a projection of a bipartite model as candidates to form the basis of a random graph model which exhibits a non-trivial number of short loops. This work was particularly motivated by the clustering exhibited by protein-protein interactions, which is believed to hold biological significance.

The Strauss model is known to suffer from clumping above certain critical values of the parameters. Below this value, we use the results of Burda et al. [2004b] to provide a leading order expression for the entropy per node. However, with reference to real biological networks, and with reference to the contribution to the entropy of the ensemble, we show that the Strauss term will only go a small way towards modelling the clustering of real networks. Using a rough Poissonian approximation, we estimate the entropy at the scalings where the number of triangles is in the order of 1 per node. We propose that the best direction for future development would be to include higher order terms to inhibit the clumping. The calculation is formulated in terms of the replica approach. The completion of the calculation would require careful analysis of the validity of taking an analytic continuation into the complex plane - this is left for future work.

We then take inspiration from the work of Agliari et al. [2013a] in order to propose a model

where the protein-protein interaction network is viewed as the projection from the bipartite relationship between complexes and proteins. There is a link between two nodes C and P in the bipartite network if protein P is a constituent of complex C. In the projection, proteins P and P' are linked if and only if they appear in the same protein. This is an attractive physical interpretation, which can be generalised for many other real-world applications. Crucially, the underlying bipartite network is tree-like, allowing the use of standard tools of statistical mechanics. The projected graph has a high proportion of short loops, typically appearing as cliques. The entropy is estimated with a heuristic argument, and the calculation is once more phrased within the framework of the replica approach.

Both models offer strong potential to yield to rigorous analytical treatment. They also each offer an exciting degree of flexibility in the features of the clustering topology which can be controlled.

CHAPTER 6

UNBIASED DEGREE-PRESERVING RANDOMISATION OF DIRECTED BINARY NETWORKS

The preceding chapters have studied various constraints that can be applied to define random graph ensembles. Several new results have now been presented which allow the calculation of the Shannon Entropy under a much wider range of previous constraints than was previously possible. To bridge these results to molecular biology it is necessary to have not only the formal definitions of these ensembles, but also methods to use these model networks in order to evaluate, contextualise or predict biological observations.

When seeking to assess the statistical relevance of observations made in real networks, there are three routes available. One could generate null-model networks for hypothesis testing from scratch, constrained by the values of observed parameters in the real network (e.g. using the Molloy and Reed [1995] stub-joining method, or the Barabási and Albert [1999] preferential attachment model). Alternatively, one could generate null-model networks by randomising the original network, using dynamical rules that leave the values of relevant parameters invariant

Rao et al. [1996]. The final option is to use analytical methods to find ensemble averages for the observable of interest, see e.g. Squartini and Garlaschelli [2011a], Squartini et al. [2011], Squartini and Garlaschelli [2011b], Del Genio et al. [2010].

The null-model approach is appealing in its conceptual simplicity. It effectively provides synthetic 'data', which can be analysed in the same way as the real dataset. One can then learn which observed properties are particular to the real dataset, and which are common within the ensemble. In this chapter we will develop some new results on a particular method of generating unbiased directed networks with constrained topologies.

Applications of network null-models are wide ranging and central to network science. Shen-Orr et al. [2002] applies null-models to identify over-represented 'motifs' in the transcriptional regulation network of *E. coli.* Pagani and Aiello [2013] discusses adapting the Watts-Strogartz method to generating random networks to model power grids. Ohnishi et al. [2010] explores motifs found within an interfirm network. Newman [2002] uses network null-models to study social networks. Maslov and Sneppen [2002] compares topological properties of interaction and transcription regulatory networks in yeast with randomised null-model networks and postulates that links between highly connected proteins are suppressed in protein interaction networks. Gotelli and Ulrich [2012] discusses the challenges of specifying a suitable matrix null-model in the field of ecology.

It is crucial that the synthetic networks generated as null-models are representative of the underlying ensembles. Any inadvertent bias in the network generation process may invalidate the hypothesis test. It is therefore worrying that the two most popular methods to randomise or generate null networks are often implemented in a biased way. That is, the processes do not reach every valid network in the ensemble with equal probability. The common implementation of the stub-joining method, where invalid edges are rejected but the process subsequently continues (as opposed to starting from the beginning), is known to be biased (e.g. see King [2004], Klein-Hennig and Hartmann [2012], Kim et al. [2012]). Even if upon invalid edge rejection the stub-joining process is restarted, it is not clear whether the graphs produced would be unbiased (we are not aware of any published proof). Del Genio et al. [2010] proposes a version of the stub-joining algorithm with an analytical correction to network observables. Similarly, the conventional 'accept-all' edge swap process, see e.g. Itzkovitz et al. [2003], is also known to be biased (e.g. see Coolen et al. [2009]): graphs on which many swaps can be executed

are generated more often. The effects of these biases may in the past not always have been serious (e.g. see Milo et al. [2004]), but using biased algorithms for producing null-models is fundamentally unsound, and unacceptable when there are rigorous unbiased alternatives.

In this chapter we build on the work of Coolen et al. [2009] and Rao et al. [1996] and define a Markov Chain Monte Carlo (MCMC) process, based on ergodic in- and out- degree preserving edge-swap moves that act on *directed* networks. We first calculate correct move acceptance probabilities for the process to sample the space of all allowed directed graphs uniformly. We then extend the theory in order for the process to evolve to any desired measure on the space of directed graphs. Attention is paid to adapting our results for efficient numerical implementation. We also identify under which circumstances the error made by doing 'accept all' edge swaps is immaterial. We apply our theory to real and synthetic networks.

6.1 An ergodic and unbiased randomisation process which preserves in- and out-degrees

6.1.1 Edge swap moves

The canonical moves for degree-preserving randomisation of graphs are the so-called 'edge swaps', see e.g. Seidel [1973], Taylor [1981], Coolen et al. [2009]. The undirected version of the edge swap is illustrated in figure 6.1; a generalisation to directed graphs is found in Rao et al. [1996]. Rao et al. [1996] define a move - which we will refer to as a *square swap* - starting from a set of four entries from the connectivity matrix $\boldsymbol{c} \in \{0,1\}^{N^2}$ of a directed binary N-node graph, defined by node pairs $\{(i_1, j_1), (i_1, j_2), (i_2, j_2), (i_2, j_1)\}$ such that the corresponding entries $\{c_{i_1 j_1}, c_{i_1,j_2}, c_{i_2,j_2}, c_{i_2,j_1}\}$ are alternately 0 and 1, and not 'structural' (i.e. they are allowed to vary). As for the undirected case, the elementary edge swap move is defined by swapping the 0 and 1 entries, i.e. $\{c_{i_1 j_1}, c_{i_1,j_2}, c_{i_2,j_2}, c_{i_2,j_1}\} \to \{1{-}c_{i_1 j_1}, 1{-}c_{i_1,j_2}, 1{-}c_{i_2,j_2}, 1{-}c_{i_2,j_1}\}$. It is built into the definition that a move is not allowed if the pair of nodes to which an edge is rewired is already connected. Rao et al. [1996] prove that, if self-interactions are permitted, repeated application of such moves can transform any binary matrix $\boldsymbol{c}_A$ to any other binary matrix $\boldsymbol{c}_B$ with the same in- and out- degree distributions. However, if we require in addition that the diagonal elements of all $\boldsymbol{c}$ are 0 (i.e.we forbid self-interactions), then the edge swap defined above is no longer sufficient to ensure ergodicity. To remedy this problem Rao et al. [1996] introduce a further

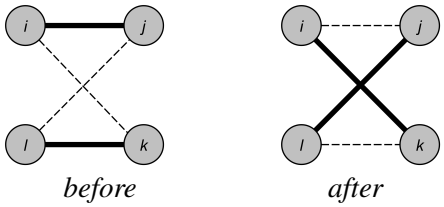


Figure 6.1: The undirected edge swap. This is the canonical choice for the elementary moves of an ergodic degree-preserving randomisation process on undirected networks.

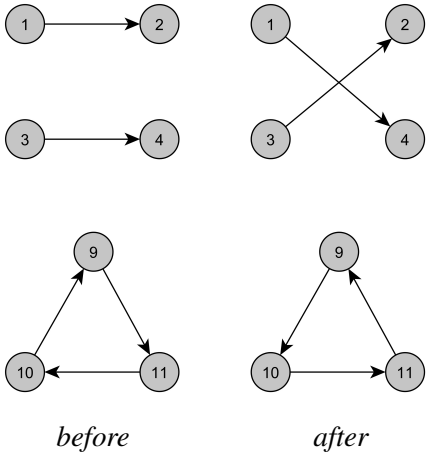


Figure 6.2: The *square swap* (top) and *triangle swap* (bottom). In combination these two represent the canonical choice for the elementary moves of an ergodic degree-preserving randomisation process on directed networks without self-interactions.

move, which we will call a *triangle swap*. This move also gives us the simplest demonstration of two valid configurations that cannot be connected by square-type swaps. The *square swap* and the *triangle swap* are illustrated in figure 6.2; in combination these two moves allow us to transform between any two directed binary matrices which have the same in- and out-degrees, even if self-interactions are forbidden Rao et al. [1996].

A stochastic process defined as accepting all randomly selected moves from the above set is ergodic but biased. This was already observed in Rao et al. [1996], where the authors proposed a 'Switch & Hold' algorithm, which involves the number of states accessible in one move from a configuration (its mobility), and the maximum possible number of states accessible in one move from any network in the ensemble (the degrees of a hyper-graph, in the language of later publications). In Coolen et al. [2009] the problem was studied for undirected graphs; it was

shown how move acceptance probabilities should be defined to guarantee stochastic evolution by edge swapping to any desired measure on the space of nondirected graphs. The analysis in Coolen et al. [2009] is quite general, and briefly reproduced in section 6.1.2 below. The new result presented in this chapter is the generalisation to directed graphs and include the new moves defined by Rao et al. [1996]. This will result in a Markovian process based on edge swapping that will equilibrate to any desired measure on the space of *directed* graphs.

6.1.2 Outline of the general theory

This section briefly summarizes results of Coolen et al. [2009] which will be used in the next section. We define an adjacency matrix $\boldsymbol{c} = \{c_{ij}\}$, where $c_{ij} = 1$ if and only if there is a directed link from node j to node i. We denote the set of all such graphs as C. The aim is to define and study constrained Markov chains for the evolution of $\boldsymbol{c}$ in some subspace $\Omega \in C$. This is a discrete time stochastic process, where the probability $p_t(\boldsymbol{c})$ of observing a graph $\boldsymbol{c}$ at time t evolves according to

$$\forall \boldsymbol{c} \in \Omega \quad : \quad p_{t+1}(\boldsymbol{c}) = \sum_{\boldsymbol{c}' \in \Omega} W(\boldsymbol{c}|\boldsymbol{c}') p_t(\boldsymbol{c}') \tag{6.1.1}$$

where $t \in \mathbb{N}$ and $W(\boldsymbol{c}|\boldsymbol{c}')$ is a transition probability. We require the process to have three additional properties:

1. Each elementary move F can only act on a subset of all possible graphs.
2. The process converges to the invariant measure $p_\infty(\boldsymbol{c}) = Z^{-1} e^{-H(\mathbf{c})}$ where Z is a normalising constant and $H(\boldsymbol{c})$ is called the Hamiltonian.
3. Each move F has a unique inverse, which acts on the same subset of states as F itself.

The second property is the crucial one, in that it defines the equilibrium probability distribution of $\boldsymbol{c}$ after the Markov process has equilibrated. For example, aiming for a flat distribution is equivalent to requiring $H(\boldsymbol{c})$ to be constant.

We exclude the identity move from the set Φ of all moves, and we define an indicator function $I_F(\boldsymbol{c})$ where $I_F(\boldsymbol{c}) = 1$ if and only if the move $\boldsymbol{c} \to F\boldsymbol{c}$ is allowed. The transition probabilities

are constructed to obey detailed balance

$$\forall \boldsymbol{c}, \boldsymbol{c}' \in \Omega \quad : \quad W(\boldsymbol{c}|\boldsymbol{c}')p_\infty(\boldsymbol{c}') = W(\boldsymbol{c}'|\boldsymbol{c})p_\infty(\boldsymbol{c}) \tag{6.1.2}$$

At each step a candidate move $F \in \Phi$ is drawn with probability $q(F|\boldsymbol{c}')$, where $\boldsymbol{c}'$ is the current state. The move is accepted with some probability $A(F\boldsymbol{c}'|\boldsymbol{c}')$. Our aim is to define a suitable $A(F\boldsymbol{c}'|\boldsymbol{c}')$ in a way such that our process will achieve the desired equilibrium distribution $p_\infty(\boldsymbol{c})$.

The relationship between the transition probability and the probability of drawing and accepting a particular move is clearly

$$W(\boldsymbol{c}|\boldsymbol{c}') \quad = \quad \sum_{F\in\Phi} q(F|\boldsymbol{c}') \left[\delta_{\mathbf{c},F\mathbf{c}'} A(F\boldsymbol{c}'|\boldsymbol{c}') + \delta_{\mathbf{c},\mathbf{c}'} \left[1 - A(F\boldsymbol{c}'|\boldsymbol{c}')\right]\right] \tag{6.1.3}$$

Insertion into (6.1.2) then leads to the following condition which must be satisfied by $A(F\boldsymbol{c}'|\boldsymbol{c}')$ and $q(F|\boldsymbol{c}')$

$$(\forall \boldsymbol{c} \in \Omega)(\forall F \in \Phi) : \tag{6.1.4}$$

$$q(F|\boldsymbol{c})A(F\boldsymbol{c}|\boldsymbol{c})e^{-H(\mathbf{c})} = q(F^{-1}|F\boldsymbol{c})A(\boldsymbol{c}|F\boldsymbol{c})e^{-H(F\mathbf{c})}$$

We define the mobility $n(\boldsymbol{c})$ to be the number of moves which can act on each state: $n(\boldsymbol{c}) = \sum_{F\in\Phi} I_F(\boldsymbol{c})$. If the candidate moves are drawn randomly with equal probabilities from the set of all moves allowed to act, we find that (6.1.4) reduces to

$$A(\boldsymbol{c}|\boldsymbol{c}') = \frac{n(\boldsymbol{c}')e^{-\frac{1}{2}[H(\mathbf{c})-H(\mathbf{c}')]}}{n(\boldsymbol{c}')e^{-\frac{1}{2}[H(\mathbf{c})-H(\mathbf{c}')]} + n(\boldsymbol{c})e^{\frac{1}{2}[H(\mathbf{c})-H(\mathbf{c}')]}} \tag{6.1.5}$$

If we make the simplest choice $H(\boldsymbol{c}) = const$, the above process will asymptotically sample all graphs with the imposed degree sequence uniformly. To sample this constrained space of graphs with alternative nontrivial probabilities $p_\infty(\boldsymbol{c})$ we would choose $H(\boldsymbol{c}) = -\log p_\infty(\boldsymbol{c}) + const$.

Equation 6.1.4 also shows what would happen if we were to sample with $A(\boldsymbol{c}|\boldsymbol{c}') \equiv 1$ for all $(\boldsymbol{c}, \boldsymbol{c}')$, i.e. for 'accept all' edge swapping: the detailed balance condition would give

$$(\forall \boldsymbol{c} \in \Omega)(\forall F \in \Phi) : \qquad \frac{e^{-H(\mathbf{c})}}{n(\boldsymbol{c})} = \frac{e^{-H(F\mathbf{c})}}{n(F\boldsymbol{c})} \tag{6.1.6}$$

For this to be satisfied we require both sides of the expression to evaluate to a constant. Hence $e^{-H(\mathbf{c})} \propto n(\boldsymbol{c})$, so the naive process will converge to the non-uniform measure

$$p_\infty(\boldsymbol{c}) = Z^{-1} n(\boldsymbol{c}) \tag{6.1.7}$$

This is the undesirable bias of 'accept-all' edge-swapping. It has a clear intuitive explanation. The mobility $n(\boldsymbol{c})$ is the number of allowed moves on network $\boldsymbol{c}$, which is equal to the number of inverse moves through which $\boldsymbol{c}$ can be reached in one step from another member of the ensemble. The likelihood of seeing a network $\boldsymbol{c}$ upon equilibration of the process is proportional to the number of entry points that $\boldsymbol{c}$ offers the process.

6.1.3 Calculation of the mobilities for directed networks

Since the two types of moves required for ergodic evolution of directed graphs - the *square swap* and the *triangle swap* - are clearly distinct, the mobility of a graph $\boldsymbol{c}$ is given by $n(\boldsymbol{c}) = n_{\square}(\boldsymbol{c}) + n_{\triangle}(\boldsymbol{c})$, where $n_{\square}(\boldsymbol{c})$ and $n_{\triangle}(\boldsymbol{c})$ count the number of possible moves of each type that can be executed on $\boldsymbol{c}$.

To find $n_{\square}(\boldsymbol{c})$ we need to calculate how many distinct link-alternating cycles of length 4 can be chosen in graph $\boldsymbol{c}$. We exclude self-interactions, so our cycles must involve 4 distinct nodes. The total number of such moves can be written as

$$n_{\square}(\boldsymbol{c}) = \frac{1}{2}\sum_{ijk\ell} \bar{\delta}_{jk}\bar{\delta}_{\ell i}\bar{\delta}_{ik}\bar{\delta}_{j\ell}c_{ij}c_{k\ell}\bar{c}_{kj}\bar{c}_{i\ell} \tag{6.1.8}$$

where the pre-factor compensates for the symmetry, and where we used the shorthand $\bar{c}_{kj} = 1 - c_{kj}$ and $\bar{\delta}_{jk} = 1 - \delta_{jk}$. Expanding this shorthand in (6.1.8) gives after some further bookkeeping of terms, and with $(\boldsymbol{c}^{\dagger})_{ij} = c_{ji}$:

$$\begin{aligned} n_{\square}(\boldsymbol{c}) &= \frac{1}{2}\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) - \sum_{ij} k_i^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}} + \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) + \frac{1}{2}\mathrm{Tr}(\boldsymbol{c}^2) \\ &\quad + \frac{1}{2}N^2\langle k\rangle^2 - \sum_j k_j^{\mathrm{in}}k_j^{\mathrm{out}} \end{aligned} \tag{6.1.9}$$

with $\langle k\rangle = N^{-1}\sum_i k_i^{\mathrm{in}} = N^{-1}\sum_i k_i^{\mathrm{out}}$. We next repeat the calculation for the case of the *triangle swap*. For easier manipulations, we introduce a new matrix $\mathbf{c}^{\updownarrow}$ of double links, defined via $(\boldsymbol{c}^{\updownarrow})_{ij} = c_{ij}c_{ji}$. We then find

$$\begin{aligned} n_{\triangle}(\boldsymbol{c}) &= \frac{1}{3}\sum_{ijk} \bar{\delta}_{ij}\bar{\delta}_{jk}\bar{\delta}_{ki}c_{ij}c_{jk}c_{ki}\bar{c}_{ji}\bar{c}_{kj}\bar{c}_{ik} \\ &= \frac{1}{3}\Big\{\mathrm{Tr}(\boldsymbol{c}^3) - 3\mathrm{Tr}(\boldsymbol{c}^{\updownarrow}\boldsymbol{c}^2) + 3\mathrm{Tr}(\boldsymbol{c}^{\updownarrow 2}\boldsymbol{c}) + -\mathrm{Tr}(\boldsymbol{c}^{\updownarrow 3})\Big\} \\ &= \frac{1}{3}\mathrm{Tr}((\boldsymbol{c} - \boldsymbol{c}^{\updownarrow})^3) \end{aligned} \tag{6.1.10}$$

In combination, (6.1.9) and (6.1.10) give us an explicit and exact formula for the graph mobility $n(\boldsymbol{c}) = n_{\square}(\boldsymbol{c}) + n_{\triangle}(\boldsymbol{c})$, and hence via (6.1.5) a fully exact MCMC process for generating random graphs with prescribed sequences and any desired probability measure in the standard form $Z^{-1}\exp[-H(\boldsymbol{c})]$.

6.2 Properties and impact of graph mobility

6.2.1 Bounds on the mobility

We will now derive bounds on the sizes of the mobility terms. This will help us characterise degree distributions for which the error due to 'accept all' edge swapping is not expected to be material.

We first observe that

$$n_{\triangle}(\boldsymbol{c}) = \frac{1}{3}\sum_{ijk} c_{ij}(1-c_{ji})c_{jk}(1-c_{kj})c_{ki}(1-c_{ik}) \leq \frac{1}{3}\mathrm{Tr}(\boldsymbol{c}^3) \tag{6.2.1}$$

Hence, the mobility $n(\boldsymbol{c}) = n_{\square}(\boldsymbol{c}) + n_{\triangle}(\boldsymbol{c})$ obeys

$$n(\boldsymbol{c}) \leq \frac{1}{2}\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) - \sum_{ij} k_i^{\mathrm{in}} c_{ij} k_j^{\mathrm{out}} + \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) + \frac{\mathrm{Tr}(\boldsymbol{c}^2)}{2} + \frac{N^2\langle k\rangle^2}{2} - \sum_j k_j^{\mathrm{in}} k_j^{\mathrm{out}} + \frac{\mathrm{Tr}(\boldsymbol{c}^3)}{3} \tag{6.2.2}$$

We find upper bounds for most of the terms above by applying the simple inequality $c_{ij}c_{kl} \leq \frac{1}{2}(c_{ij} + c_{kl})$, which gives e.g.

$$\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) \leq \frac{N}{2}\left[\langle {k^{\mathrm{in}}}^2\rangle + \langle {k^{\mathrm{out}}}^2\rangle\right] \qquad \mathrm{Tr}(\boldsymbol{c}^2) \leq N\langle k\rangle \tag{6.2.3}$$

$$\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) \leq \sum_{ij} k_i^{\mathrm{in}} c_{ij} k_i^{\mathrm{out}} \qquad \mathrm{Tr}(\boldsymbol{c}^3) \leq N\langle k^{\mathrm{in}} k^{\mathrm{out}}\rangle \tag{6.2.4}$$

An upper bound on $\sum_{ij} k_i^{\mathrm{in}} c_{ij} k_j^{\mathrm{out}}$ follows from the observation that if $c_{ij} = 1$ then certainly $k_i^{\mathrm{in}} \geq 1$ and $k_j^{\mathrm{out}} \geq 1$. Hence

$$\begin{aligned} \sum_{ij} k_i^{\mathrm{in}} c_{ij} k_j^{\mathrm{out}} &\geq \frac{1}{2}\sum_{ij}[c_{ij}k_j^{\mathrm{out}} + k_i^{\mathrm{in}} c_{ij}] \\ &= \frac{1}{2}N[\langle {k^{\mathrm{in}}}^2\rangle + \langle {k^{\mathrm{out}}}^2\rangle] \end{aligned} \tag{6.2.5}$$

Combining (6.2.3,6.2.4,6.2.5) with (6.2.2) then gives

$$n(\boldsymbol{c}) \leq \frac{N}{2}\Big[N\langle k\rangle^2 + \langle k\rangle + \frac{1}{2}[\langle {k^{\mathrm{in}}}^2\rangle + \langle {k^{\mathrm{out}}}^2\rangle] - \frac{4}{3}\langle k^{\mathrm{in}} k^{\mathrm{out}}\rangle\Big] \tag{6.2.6}$$

Next we calculate a lower bound for $n(\boldsymbol{c})$. We use simple identities such as

$$\mathrm{Tr}(\boldsymbol{c}^2) \geq 0 \qquad n_{\triangle}(\boldsymbol{c}) \geq 0 \qquad \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) \geq 0 \tag{6.2.7}$$

and

$$\begin{aligned} \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) &\geq \frac{1}{2}\sum_{ijk\ell} c_{ji}c_{jk}c_{\ell k}c_{\ell i}(\delta_{j\ell}+\delta_{ik}) \\ &= N[\langle k^{\mathrm{in}\,2}\rangle + \langle k^{\mathrm{out}\,2}\rangle] \end{aligned} \tag{6.2.8}$$

We now find

$$n(\boldsymbol{c}) \geq \tfrac{1}{2}N[\langle k^{\mathrm{in}\,2}\rangle + \langle k^{\mathrm{out}\,2}\rangle] + \tfrac{1}{2}N^2\langle k\rangle^2 - \textstyle\sum_j k_j^{\mathrm{in}}k_j^{\mathrm{out}} - \sum_{ij} k_i^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}} \tag{6.2.9}$$

We finally need an upper bound for $\sum_{ij} k_i^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}}$, which we write in terms of $k_{\mathrm{max}}^{\mathrm{in}} = \max_i k_i^{\mathrm{in}}$ and $k_{\mathrm{max}}^{\mathrm{out}} = \max_i k_i^{\mathrm{out}}$:

$$\sum_{ij} k_i^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}} \leq \frac{1}{2}\sum_{ij}[k_{\mathrm{max}}^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}} + k_i^{\mathrm{in}}c_{ij}k_{\mathrm{max}}^{\mathrm{out}}] = \frac{1}{2}N\Big[k_{\mathrm{max}}^{\mathrm{in}}\langle k^{\mathrm{out}\,2}\rangle + k_{\mathrm{max}}^{\mathrm{out}}\langle k^{\mathrm{in}\,2}\rangle\Big] \tag{6.2.10}$$

We thus obtain our lower bound for the mobility:

$$n(\boldsymbol{c}) \geq \frac{N}{2}\Big[N\langle k\rangle^2 + \langle (k^{\mathrm{in}}-k^{\mathrm{out}})^2\rangle - k_{\mathrm{max}}^{\mathrm{in}}\langle k^{\mathrm{out}\,2}\rangle - k_{\mathrm{max}}^{\mathrm{out}}\langle k^{\mathrm{in}\,2}\rangle\Big] \tag{6.2.11}$$

6.2.2 Identification of graph types most likely to be biased by 'accept all' edge swapping

We know from (6.1.5) that unbiased sampling of graphs, i.e. $p(\boldsymbol{c}) = 1/|\Omega|$ for all $\boldsymbol{c} \in \Omega$ using a process with a non-constant number of possible moves at each step requires using the following state-dependent acceptance probabilities in the edge swap process:

$$A(\boldsymbol{c}|\boldsymbol{c}') = [1 + n(\boldsymbol{c})/n(\boldsymbol{c}')]^{-1} \tag{6.2.12}$$

We now investigate under which conditions one will in large graphs effectively find $n(\boldsymbol{c})/n(\boldsymbol{c}') \to 1$ for all $\boldsymbol{c}, \boldsymbol{c}' \in \Omega$, so that the sampling bias would be immaterial. Let us define

$$\Delta n = \max_{\boldsymbol{c},\boldsymbol{c}'\in\Omega} |n(\boldsymbol{c}) - n(\boldsymbol{c}')| = \max_{\boldsymbol{c}\in\Omega} n(\boldsymbol{c}) - \min_{\boldsymbol{c}\in\Omega} n(\boldsymbol{c}) \tag{6.2.13}$$

Using the two bounds (6.2.6,6.2.11) we immediately obtain

$$\begin{aligned} \Delta n \leq& \frac{N}{2}\Big[\langle k\rangle - \frac{1}{2}[\langle k^{\mathrm{in}\,2}\rangle + \langle k^{\mathrm{out}\,2}\rangle] + \frac{2}{3}\langle k^{\mathrm{in}}k^{\mathrm{out}}\rangle k_{\mathrm{max}}^{\mathrm{in}}\langle k^{\mathrm{out}\,2}\rangle + k_{\mathrm{max}}^{\mathrm{out}}\langle k^{\mathrm{in}\,2}\rangle\Big] \\ =& \frac{N}{2}\Big[\langle k\rangle - \frac{1}{6}[\langle k^{\mathrm{in}\,2}\rangle + \langle k^{\mathrm{out}\,2}\rangle] - \frac{1}{3}\langle (k^{\mathrm{in}}-k^{\mathrm{out}})^2\rangle + k_{\mathrm{max}}^{\mathrm{in}}\langle k^{\mathrm{out}\,2}\rangle + k_{\mathrm{max}}^{\mathrm{out}}\langle k^{\mathrm{in}\,2}\rangle\Big] \\ \leq& \frac{N}{2}\Big[\langle k\rangle + k_{\mathrm{max}}^{\mathrm{in}}\langle k^{\mathrm{out}\,2}\rangle + k_{\mathrm{max}}^{\mathrm{out}}\langle k^{\mathrm{in}\,2}\rangle\Big] \end{aligned} \tag{6.2.14}$$

Clearly $1 - \Delta n/n(\boldsymbol{c}) \leq n(\boldsymbol{c}')/n(\boldsymbol{c}) \leq 1 + \Delta n/n(\boldsymbol{c})$, so in view of (6.2.12) we are interested in the ratio $\Delta n/n(\boldsymbol{c})$, for which we find

$$\frac{\Delta n}{n(\boldsymbol{c})} \quad \leq \quad \frac{\langle k\rangle + k^{\rm in}_{\rm max}\langle k^{\rm out\,2}\rangle + k^{\rm out}_{\rm max}\langle k^{\rm in\,2}\rangle}{N\langle k\rangle^2 - k^{\rm in}_{\rm max}\langle k^{\rm out\,2}\rangle - k^{\rm out}_{\rm max}\langle k^{\rm in\,2}\rangle} \tag{6.2.15}$$

So we can be confident that the impact of the graph mobility on the correct acceptance probabilities (6.2.12) is immaterial if

$$\frac{1}{\langle k\rangle} + \frac{2}{\langle k\rangle^2}\Big(k^{\rm in}_{\rm max}\langle k^{\rm out\,2}\rangle + k^{\rm out}_{\rm max}\langle k^{\rm in\,2}\rangle\Big) \ll N \tag{6.2.16}$$

We see from this that we can apply the 'accept all' edge-swap process with confidence when we are working with a large network with a narrow degree distribution.

6.2.3 Simple graph examples

In this section we confirm the validity of the mobility formulae (6.1.9,6.1.10) for several simple examples of directed graphs.

1. *Two isolated bonds:*

   Here we have $c_{12} = 1$, $c_{34} = 1$, and $c_{ij} = 0$ for all $(i, j) \notin \{(1, 2), (3, 4)\}$. It is immediately clear that $\boldsymbol{c}^{\updownarrow} = \boldsymbol{0}$, and

   $$\sum_{ij} k^{\rm in}_i c_{ij} k^{\rm out}_j = 2, \quad \sum_j k^{\rm out}_j k^{\rm in}_j = 0, \quad \langle k\rangle = \frac{2}{N} \tag{6.2.17}$$
   $$\mathrm{Tr}(\boldsymbol{c}^2) = \mathrm{Tr}(\boldsymbol{c}^3) = \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}) = 0, \quad \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}\boldsymbol{c}^\dagger) = 2$$

   Insertion into (6.1.9,6.1.10) gives $n_\square(\boldsymbol{c}) = 1$ and $n_\triangle(\boldsymbol{c}) = 0$. As we would expect: only one (square) move is permitted.

2. *Isolated triangle:*

   This example is defined by $c_{12} = c_{23} = c_{31} = 1$, with $c_{ij} = 0$ for all $(i, j) \notin \{(1, 2), (2, 3), (3, 1)\}$. Again we have $\boldsymbol{c}^{\updownarrow} = \boldsymbol{0}$, but now

   $$\sum_{ij} k^{\rm in}_i c_{ij} k^{\rm out}_j = 3, \quad \sum_j k^{\rm out}_j k^{\rm in}_j = 3, \quad \langle k\rangle = \frac{3}{N} \tag{6.2.18}$$
   $$\mathrm{Tr}(\boldsymbol{c}^2) = \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}) = 0, \quad \mathrm{Tr}(\boldsymbol{c}^3) = 3, \quad \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}\boldsymbol{c}^\dagger) = 3$$

   This results in $n_\square(\boldsymbol{c}) = 0$ and $n_\triangle(\boldsymbol{c}) = 1$. The only possible move is reversal of the directed triangle.

3. *Complete (fully connected) graph:*

   Here $c_{ij} = 1 - \delta_{ij}$, and no edge swaps are possible. All nodes have $k_i^{\text{in}} = k_i^{\text{out}} = N - 1$, and since $\boldsymbol{c}^{\updownarrow} = \boldsymbol{c}$ we know that $n_{\triangle}(\boldsymbol{c}) = 0$. This connectivity matrix, also featured in Coolen et al. [2009], has eigenvalues $\lambda = N - 1$ (multiplicity 1) and $\lambda = -1$ (multiplicity $N - 1$). Hence

$$\begin{aligned}
&\sum_{ij} k_i^{\text{in}} c_{ij} k_j^{\text{out}} = N(N-1)^3, \quad \sum_j k_j^{\text{out}} k_j^{\text{in}} = N(N-1)^2 \\
&\text{Tr}(\boldsymbol{c}^2) = \sum_i \lambda_i^2 = N(N-1), \\
&\text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}) = \text{Tr}(\boldsymbol{c}^3) = \sum_i \lambda_i^3 = N(N-1)(N-2), \\
&\text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}\boldsymbol{c}^\dagger) = \text{Tr}(\boldsymbol{c}^4) = \sum_i \lambda_i^4 = (N-1)[(N-1)^3 + 1]
\end{aligned} \tag{6.2.19}$$

   Assembling the entire expression for the square mobility term (6.1.9) indeed gives the correct value $n_{\square}(\boldsymbol{c}) = 0$.

4. *Directed spanning ring:*

   This directed graph, defined by $c_{ij} = \delta_{i+1,j}$ modulo N, gives a ring with a flow around it. We choose $N > 2$. Once more $\boldsymbol{c}^{\updownarrow} = \boldsymbol{0}$, and we obtain for the relevant terms

$$\begin{aligned}
&\sum_{ij} k_i^{\text{in}} c_{ij} k_j^{\text{out}} = \sum_j k_j^{\text{out}} k_j^{\text{in}} = N, \quad \langle k \rangle = 1 \\
&\text{Tr}(\boldsymbol{c}^2) = \text{Tr}(\boldsymbol{c}^3) = \text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}) = 0, \quad \text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}\boldsymbol{c}^\dagger) = N
\end{aligned} \tag{6.2.20}$$

   The final result, $n_{\square}(\boldsymbol{c}) = \frac{1}{2}N(N-3)$ and $n_{\triangle}(\boldsymbol{c}) = 0$, is again what we would expect. As soon as one first bond to participate in an edge swap is picked (for which there are N options), there are $N - 3$ possibilities for the second (since the already picked bond and its neighbours are forbidden). The factor 2 corrects for over-counting.

5. *Bidirectional spanning ring:*

   Our final example is the nondirected version of the previous graph, viz. $c_{ij} = \delta_{i,j-1} + \delta_{i,j+1}$ modulo N, with $N > 2$. Since $\boldsymbol{c}^{\updownarrow} = \boldsymbol{c}$ we have $n_{\triangle}(\boldsymbol{c}) = 0$. Now

$$\begin{aligned}
&\sum_{ij} k_i^{\text{in}} c_{ij} k_j^{\text{out}} = 8N, \quad \sum_j k_j^{\text{out}} k_j^{\text{in}} = 4N, \quad \langle k \rangle = 2 \\
&\text{Tr}(\boldsymbol{c}^2) = 2N, \quad \text{Tr}(\boldsymbol{c}^3) = \text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}) = 0 \\
&\text{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger \boldsymbol{c}\boldsymbol{c}^\dagger) = 6N
\end{aligned} \tag{6.2.21}$$

We thereby find $n_{\square}(\boldsymbol{c}) = 2N(N-4)$. This is double the mobility evaluated in Coolen et al. [2009], since every move in the undirected version of the network corresponds to two possible moves in the directed version of the network.

6.3 A randomisation process which preserves degrees and targets degree-degree correlations

So far we applied formula (6.1.5) for the canonical acceptance probabilities for directed graph edge swapping to the problem of generating graphs with prescribed in- and out-degrees $(\boldsymbol{k}^{\text{in}}, \boldsymbol{k}^{\text{out}})$ and a uniform measure. Here we consider how to generate graphs which, in addition, display certain degree correlations. We first rewrite (6.1.5) as

$$A(\boldsymbol{c}|\boldsymbol{c}') = \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} e^{H(\mathbf{c})-H(\mathbf{c}')}\right]^{-1} \tag{6.3.1}$$

These probabilities (6.3.1) ensure the edge-swapping process evolves into the stationary state on $\Omega = \{\boldsymbol{c} \in \{0,1\}^{N^2} |\ \boldsymbol{k}^{\text{in}}(\boldsymbol{c}) = \boldsymbol{k}^{\text{in}},\ \boldsymbol{k}^{\text{out}}(\boldsymbol{c}) = \boldsymbol{k}^{\text{out}}\}$ defined by $p_{\infty}(\boldsymbol{c}) = Z^{-1}\exp[-H(\boldsymbol{c})]$. The full degree-degree correlation structure of a directed graph $\boldsymbol{c}$ is captured by the joint degree distribution of connected nodes

$$W(\vec{k}, \vec{k}'|\boldsymbol{c}) = \frac{1}{N\langle k\rangle} \sum_{ij} c_{ij}\, \delta_{\vec{k},\vec{k}_i(\mathbf{c})} \delta_{\vec{k}',\vec{k}_j(\mathbf{c})} \tag{6.3.2}$$

with $\vec{k} = (k^{\text{in}}, k^{\text{out}})$. The maximum entropy distribution on Ω, viz. all directed graphs with prescribed in- and out-degree sequences, that has the distribution (6.3.2) imposed as a soft constraint, i.e. $\sum_{\mathbf{c}\in\Omega} p(\boldsymbol{c}) W(\vec{k}, \vec{k}'|\boldsymbol{c}) = W(\vec{k}, \vec{k}')$ for all $(\vec{k}, \vec{k}')$, is

$$p(\boldsymbol{c}) = Z^{-1} \prod_i \delta_{\vec{k}_i,\vec{k}_i(\mathbf{c})} \times \prod_{ij} \left[\frac{\langle k\rangle}{N} Q(\vec{k}_i, \vec{k}_j)\delta_{c_{ij},1} + \Big(1 - \frac{\langle k\rangle}{N} Q(\vec{k}_i, \vec{k}_j)\Big)\delta_{c_{ij},0}\right] \tag{6.3.3}$$

(see Roberts et al. [2011]), in which $Q(\vec{k}, \vec{k}') = W(\vec{k}, \vec{k}')/p(\vec{k})p(\vec{k}')$ and $p(\vec{k}) = p(k^{\text{in}}, k^{\text{out}})$. It should be clear due to the context whether we are referring to $p(\boldsymbol{c})$ (the probability of observing a certain network $\boldsymbol{c}$ in the ensemble) or $p(\vec{k})$ (the probability of finding a node within a network with k^{in} incoming and k^{out} outgoing edges). It is now trivial, following Coolen et al. [2009], to ensure that our MCMC process evolves to the measure (6.3.3) by choosing $H(\boldsymbol{c}) = -\log p(\boldsymbol{c})$

in the probabilities (6.3.1). This gives

$$A(\boldsymbol{c}|\boldsymbol{c}') = \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \prod_{ij} \frac{\frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j) c'_{ij} + \left(1 - \frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j)\right)(1-c'_{ij})}{\frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j) c_{ij} + \left(1 - \frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j)\right)(1-c_{ij})}\right]^{-1}$$
$$= \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \prod_{ij} \left(\frac{\frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j)}{1 - \frac{\langle k\rangle}{N} Q(\vec{k}_i,\vec{k}_j)}\right)^{c'_{ij} - c_{ij}}\right]^{-1} \tag{6.3.4}$$

If the proposed move is a *square swap*, it is characterized by four distinct nodes (i, j, k, ℓ), and takes us from a graph $\boldsymbol{c}'$ with $c'_{ij}c'_{k\ell}\bar{c}'_{kj}\bar{c}'_{i\ell} = 1$ to a new graph $\boldsymbol{c}$ with $\bar{c}_{ij}\bar{c}_{k\ell}c_{kj}c_{i\ell} = 1$ (leaving all other $N^2 - 4$ bond variables unaffected). For such moves the acceptance probabilities (6.3.4) become

$$A_{\square}(\boldsymbol{c}|\boldsymbol{c}') = \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \frac{\left(\frac{N}{\langle k\rangle Q(\vec{k}_k,\vec{k}_j)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_i,\vec{k}_\ell)} - 1\right)}{\left(\frac{N}{\langle k\rangle Q(\vec{k}_i,\vec{k}_j)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_k,\vec{k}_\ell)} - 1\right)}\right]^{-1} \tag{6.3.5}$$

For large N we may choose to approximate this by

$$A_{\square}(\boldsymbol{c}|\boldsymbol{c}') \approx \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \frac{Q(\vec{k}_i,\vec{k}_j) Q(\vec{k}_k,\vec{k}_\ell)}{Q(\vec{k}_k,\vec{k}_j) Q(\vec{k}_i,\vec{k}_\ell)}\right]^{-1} \tag{6.3.6}$$

If the proposed move is a *triangle edge swap*, it is characterized by three distinct nodes (i, j, k), and takes us from a graph $\boldsymbol{c}'$ with $c'_{ij}c'_{jk}c'_{ki}\bar{c}'_{ji}\bar{c}'_{kj}\bar{c}'_{ik} = 1$ to a new graph $\boldsymbol{c}$ with $\bar{c}_{ij}\bar{c}_{jk}\bar{c}_{ki}c_{ji}c_{kj}c_{ik} = 1$ (leaving all other $N^2 - 6$ bond variables unaffected). Now the acceptance probabilities (6.3.4) become

$$A_{\triangle}(\boldsymbol{c}|\boldsymbol{c}') = \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \frac{\left(\frac{N}{\langle k\rangle Q(\vec{k}_j,\vec{k}_i)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_k,\vec{k}_j)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_i,\vec{k}_k)} - 1\right)}{\left(\frac{N}{\langle k\rangle Q(\vec{k}_i,\vec{k}_j)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_j,\vec{k}_k)} - 1\right)\left(\frac{N}{\langle k\rangle Q(\vec{k}_k,\vec{k}_i)} - 1\right)}\right]^{-1} \tag{6.3.7}$$

For large N we may choose to approximate this by

$$A_{\triangle}(\boldsymbol{c}|\boldsymbol{c}') = \left[1 + \frac{n(\boldsymbol{c})}{n(\boldsymbol{c}')} \frac{Q(\vec{k}_i,\vec{k}_j) Q(\vec{k}_j,\vec{k}_k) Q(\vec{k}_k,\vec{k}_i)}{Q(\vec{k}_j,\vec{k}_i) Q(\vec{k}_k,\vec{k}_j) Q(\vec{k}_i,\vec{k}_k)}\right]^{-1} \tag{6.3.8}$$

6.4 Numerical simulations of the canonical randomization process

In this section we describe numerical simulations of our canonical MCMC graph randomization process and its 'accept all' edge swapping counterpart, applied to synthetic networks and to biological signalling networks. We used the Mersenne Twister random number generator from

Matsumoto and Nishimura [1998]. For numerical implementation, we use expressions for the incremental change in the mobility terms following a single edge swap move (similar to how this was done for nondirected networks Coolen et al. [2009]) – see appendix D. This avoids having to calculate $n(\boldsymbol{c})$ after each move, which would involve repeated matrix multiplications. We find that the increase in running time relative to the naïve approach using our adjusted acceptance probability is not material. For example, in the real biological examples shown in figures 6.8 and 6.9, both approaches equilibrate within minutes using a desktop PC. The difference in running times between the two is less than 10%. Targeting a specific degree-degree correlation does carry a significant penalty in terms of computer time, due to needing to calculate a more complicated acceptance probability at every step. However, it is likely that the implementation could be substantially speeded up and optimised if required for a real application.

We find that the most convenient marker of sampling bias in randomisation is the mobility $n(\boldsymbol{c})$ itself, which we will therefore use to monitor the dynamics of the process. For the synthetic networks discussed in sections 6.4.1 and 6.4.2 we can calculate the mobility for each type of network, hence we can directly relate it to the proportion of time that the process actually spends in each configuration, versus the expected proportion of time for an unbiased process. For the real biological networks, since our postulate is that the 'biased' process favours networks with higher mobilities, it seems reasonable to assume that the (running) average mobility over the course of the process will be the statistic that most clearly illustrates the difference between the two properties.

6.4.1 'Split flow' network

A 'split flow' network, see e.g. Milo et al. [2004], is built as follows. Node $i = 1$ has degrees $(k_1^{\text{in}}, k_1^{\text{out}}) = (0, K)$, we have K nodes ($i = 2 \ldots K + 1$) with degrees $(k_i^{\text{in}}, k_i^{\text{out}}) = (1, 1)$, and a final node with degrees $(k_{K+2}^{\text{in}}, k_{K+2}^{\text{out}}) = (K, 0)$. There exist two types of graph with this specified degree sequence. The first is shown in the left of figure 6.3. The second type is obtained from the first by choosing two of the K 'inner nodes', of which one will cease to receive a link from $i = 1$ and the second will cease to provide a link to $i = K + 2$; so the mobility of the left graph is $n(\boldsymbol{c}) = K(K - 1)$. On the right-hand side configurations in figure 6.3 we can execute three possible square edge swap types: returning to the previous state (1 such move), changing the internal node that is not receiving a link from $i = 1$ ($K - 2$ such moves), or changing the internal

node that is not sending a link to $i = K + 2$ ($K - 2$ such moves), giving a total mobility for the graphs on the right of $n(\boldsymbol{c}) = 2K - 3$. The total number of such 'split flow' networks is $|\Omega| = K(K-1) + 1$.

Figure 6.4 shows graph randomisation dynamics for a split-flow network with $K = 25$, comparing 'accept all' edge swapping (which would sample graphs with the bias $p(\boldsymbol{c}) = n(\boldsymbol{c}) / \sum_{\boldsymbol{c}' \in \Omega} n(\boldsymbol{c}')$) to the canonical edge swap process (6.2.12) that is predicted to give unbiased sampling of graphs $p(\boldsymbol{c}) = 1/|\Omega|$. The predicted expectation values of the mobilities in the two sampling protocols are

$$\begin{aligned} \text{'accept all'}: \quad \langle n(\mathbf{c}) \rangle &= \frac{\sum_{\mathbf{c} \in \Omega} n^2(\mathbf{c})}{\sum_{\mathbf{c} \in \Omega} n(\mathbf{c})} = \frac{5K^2 - 13K + 9}{2(K-1)} \approx 58.52 \\ \text{canonical}: \quad \langle n(\boldsymbol{c}) \rangle &= \frac{\sum_{\mathbf{c} \in \Omega} n(\mathbf{c})}{|\Omega|} = \frac{2K(K-1)^2}{1 + K(K-1)} \approx 47.92 \end{aligned} \tag{6.4.1}$$

The simulation results confirm these quantitative predictions (see caption of figure 6.4 for details), and underline the sampling bias caused by 'accept all' edge swapping, as well as the lack of such a bias in our canonical MCMC process.

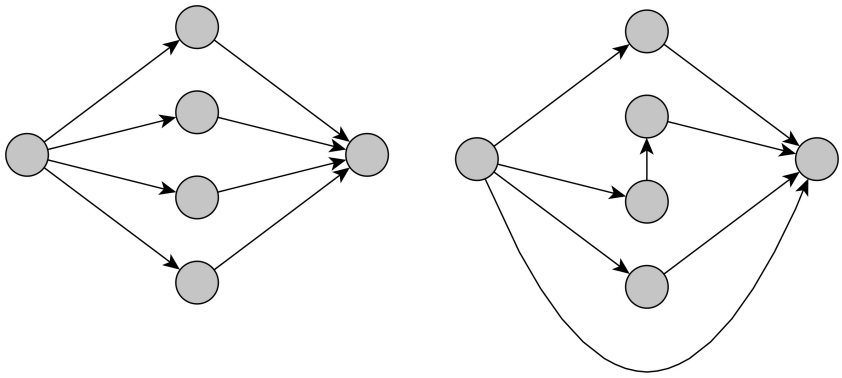


Figure 6.3: The possible realisations of a 'split flow' type network, with $N = K + 2$. The left hand configuration has a mobility of $K(K-1)$; there is only one such configuration. The right hand configuration has mobility of $2K - 3$; there are $K(K-1)$ such configurations.

6.4.2 'Nearly hardcore' networks

'Nearly hardcore' networks are another example of graphs for which 'accept all' edge swap sampling is known to exhibit a significant bias Coolen et al. [2009]. The directed version of

such networks is constructed from a single isolated bond plus a complete subgraph of size $K = N - 2$. See figure 6.5. Triangle swaps are not possible. From the graph shown in the figure (the 'mobile' state, A) there are $K(K-1)$ ways to choose two nodes of the core to combine with the two non-core nodes to form an edge swap quartet, hence this state has $n_A(\boldsymbol{c}) = K(K-1)$. After an edge swap the graph in figure 6.5 is replaced by one in which one non-core node receives a link from the core, and the other sends a link to the core; see figure 6.6. There are $K(K-1)$ such graphs, to be called type B, hence the total number of 'nearly hardcore' graphs is $|\Omega| = K(K-1)+1$. From each type B graph the inverse swap can be applied, plus $2(K-2)$ further moves that each equate to replacement of one of the core nodes involved in the previous swap by another. Hence $n_B(\boldsymbol{c}) = 2K-3$. These statements are confirmed by formula (6.1.9).

The predicted expectation values of the mobilities in the two sampling protocols, 'accept all' edge swapping (which would sample graphs with the bias $p(\boldsymbol{c}) = n(\boldsymbol{c})/\sum_{\boldsymbol{c}'\in\Omega} n(\boldsymbol{c}')$) and the canonical edge swap process (6.2.12) (predicted to give unbiased sampling of graphs $p(\boldsymbol{c}) = 1/|\Omega|$), are

$$\begin{aligned} \text{'accept all'}: \qquad \langle n(\boldsymbol{c})\rangle &= \frac{n_A^2(\boldsymbol{c})+K(K-1)n_B^2(\boldsymbol{c})}{n_A(\boldsymbol{c})+K(K-1)n_B(\boldsymbol{c})} = \frac{5K^2-13K+9}{2(K-1)} \\ \text{canonical}: \qquad \langle n(\boldsymbol{c})\rangle &= \frac{n_A(\boldsymbol{c})+K(K-1)n_B(\boldsymbol{c})}{1+K(K-1)} = \frac{2K(K-1)^2}{1+K(K-1)} \end{aligned}$$

Figure 6.7 shows graph randomisation dynamics for a 'nearly hardcore' network with $K = 18$ (so $N = 20$). Here the theory, i.e. the previous two formulae, predicts that we should see $\langle n(\boldsymbol{c})\rangle \approx 41.03$ for 'accept all' edge swapping, and $\langle n(\boldsymbol{c})\rangle \approx 33.89$ for unbiased sampling. Again the simulation results confirm our predictions (see caption of figure 6.7 for details).

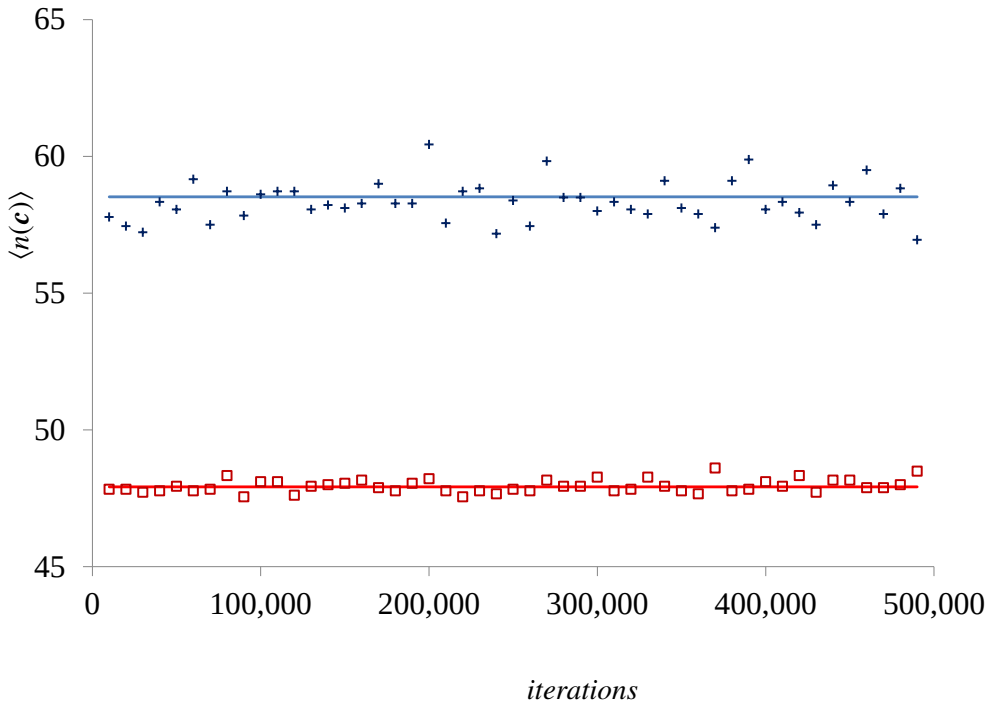


Figure 6.4: Comparison for 'split flow' networks with $K = 25$ of randomization via 'accept all' edge swapping (squares) versus edge swapping with the canonical acceptance probabilities (crosses). The mobility $\langle n(\boldsymbol{c})\rangle$ is used as a dynamical observable, since its expectation value is sensitive to sampling bias. Each marker gives the average mobility over 10,000 iterations. Observed values are in good agreement with theoretical predictions: $\langle n(\boldsymbol{c})\rangle \approx 58.32$ for 'accept all' edge swapping (predicted: 58.52 shown by top solid line), versus $\langle n(\boldsymbol{c})\rangle \approx 47.95$ for correct edge swapping (predicted: 47.92 shown by lower solid line).

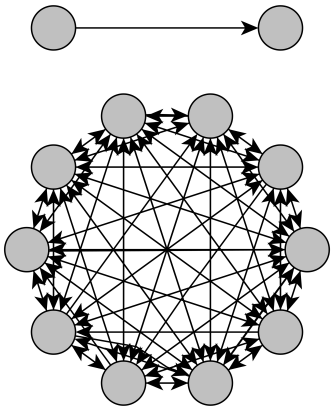


Figure 6.5: The directed version of a 'nearly hardcore' network. Given the imposed degree sequences, there are only two types of graphs: the one shown here, and the one obtained by via an edge swap that involves the nodes of the isolated link and two nodes from the core.

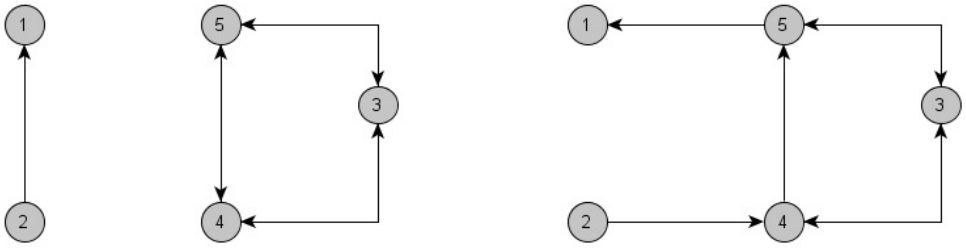


Figure 6.6: Illustration of the edge swap that transforms a 'nearly hardcore' graph from state A to one of the type B states.

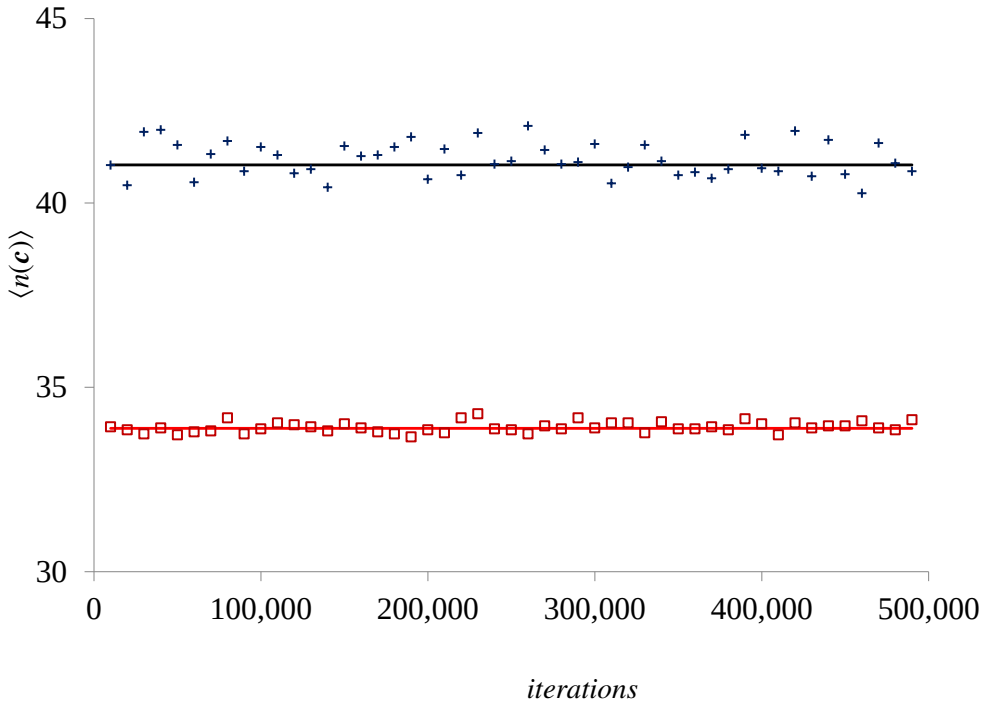


Figure 6.7: Comparison for 'nearly hardcore' networks with $K = 18$ of randomization via 'accept all' edge swapping (squares) versus edge swapping with the canonical acceptance probabilities (crosses). Each marker gives the average mobility over 10,000 iterations. Observed mobility values are again in good agreement with theoretical predictions: $\langle n(\boldsymbol{c})\rangle \approx 41.09$ for 'accept all' (predicted: 41.03 shown by top solid line), versus $\langle n(\boldsymbol{c})\rangle \approx 33.92$ for correct edge swapping (predicted: 33.89 shown by the lower solid line).

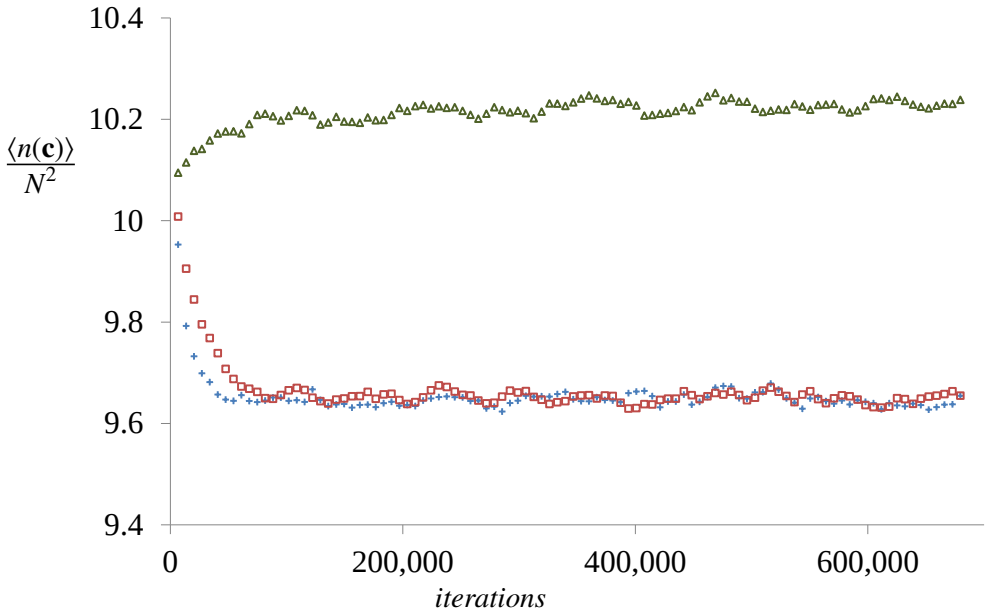


Figure 6.8: Randomization dynamics for the gene regulation network data of Hughes et al. [2000]. The observable shown is a running average of the normalized average square mobility $\langle n_{\square}(\boldsymbol{c})\rangle/N^2$. We compare 'accept all' edge swapping (+), canonical edge swapping aimed at uniform sampling of all graphs with the biological degree sequence of the biological network (□), and canonical edge swapping aimed at uniform sampling of all graphs with the degree sequence $(\vec{k}_1,\ldots,\vec{k}_N)$ *and* the degree-degree correlation kernel $W(\vec{k},\vec{k}')$ of the biological network (△).

6.4.3 Application to gene regulation networks

Gene regulation networks are important examples of directed biological networks. Figures 6.8 and 6.9 show numerical results of the randomization dynamics applied to the gene regulation network data of Hughes et al. [2000] (with $N = 5654$ nodes) and Harbison et al. [2004] (with $N = 3865$ nodes), respectively. We apply all three randomization processes discussed so far in this chapter, viz. 'accept all' edge swapping, canonical edge swapping aimed at uniform sampling of all graphs with the degree sequences of the biological network, and canonical edge swapping aimed at uniform sampling of all graphs with the degree sequence $(\vec{k}_1,\ldots,\vec{k}_N)$ *and* (on average) the degree-degree correlation kernel $W(\vec{k},\vec{k}')$ of the biological network.

To demonstrate that the above processes are effectively shuffling the networks, we measure the

Hamming distance between the start and end network. The Hamming distance is defined as $\frac{1}{2E}\sum_{ij}|c_{ij}^{start} - c_{ij}^{end}|$ where E is the total number of edges. A value of zero would indicate that the start and end networks were identical. A value of 1 would indicate that the start and end networks had no edges in common. Hamming distances between the start and end networks of □, + and △ process for figure 6.8 were 0.8, 0.8 and 0.75 respectively. Hamming distances for the process illustrated in figure 6.9 between the start and end networks of □, + and △ were 0.94, 0.94 and 0.86 respectively. Please also see figure 6.10 for a demonstration of how the △ process effectively targets the degree-degree correlation of the original network.

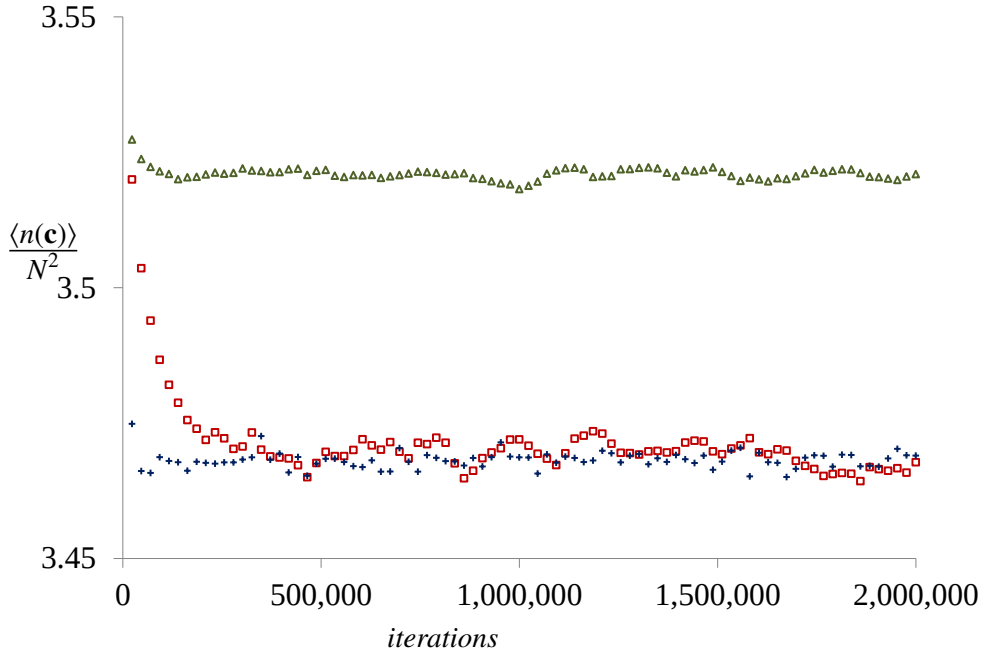


Figure 6.9: Randomization dynamics for the gene regulation network of Harbison et al. [2004]. The key and the axes are the same as in figure 6.8.

Figure 6.10 gives a visualisation of the degree-degree correlation observed in the original network of Harbison et al. [2004] versus sample randomised networks, randomised respectively to preserve degree distribution or degree-degree correlation. This average nearest neighbour degree projection was chosen as a widely adopted and easy to interpret measure of the assortativity of a directed network. The purpose of these charts is to demonstrate that the process targeting degree-degree correlations successfully reproduces the key features of the assortativity of the real networks. In particular, the pronounced downwards slope in the last four charts brings to mind the work of Maslov and Sneppen [2002] on the related problem of protein interaction

networks. Maslov and Sneppen [2002] also observed a characteristic downwards slope to the assortativity charts from their data, and postulated that this may be a key 'design' feature of real networks, contributing greater stability and improved specificity.

In contrast to the synthetic examples in the previous subsection, in gene regulation networks we do not observe significant divergence between 'accept all' versus canonical edge swap randomization; this is similar to what was observed earlier for the randomization of protein-protein interaction networks in Coolen et al. [2009]. We also see that in both cases the biological network is significantly more mobile than the typical network with the same degree sequence. However, figures 6.8 and 6.9 suggests that the set of networks that share with the biological one both the degree sequence *and* the degree correlations (and hence resemble more closely the biological network under study) all have high mobilities.

Implementing degree-degree correlation targeting directly has the effect of severely reducing the space of graphs through which the process can pass, hence we would expect finite-size effects to be more pronounced. The process would be less restricted, and hence more natural, with a smoothed target degree-degree correlation. There is a trade-off between the flexibility of the process and the accuracy of the targeting. We have used a light Gaussian smoothing, generalising what was used in Fernandes et al. [2010] to the higher dimension we need. The best choice target degree-degree correlation - including decisions about smoothing - will very much depend on the particular problem being studied.

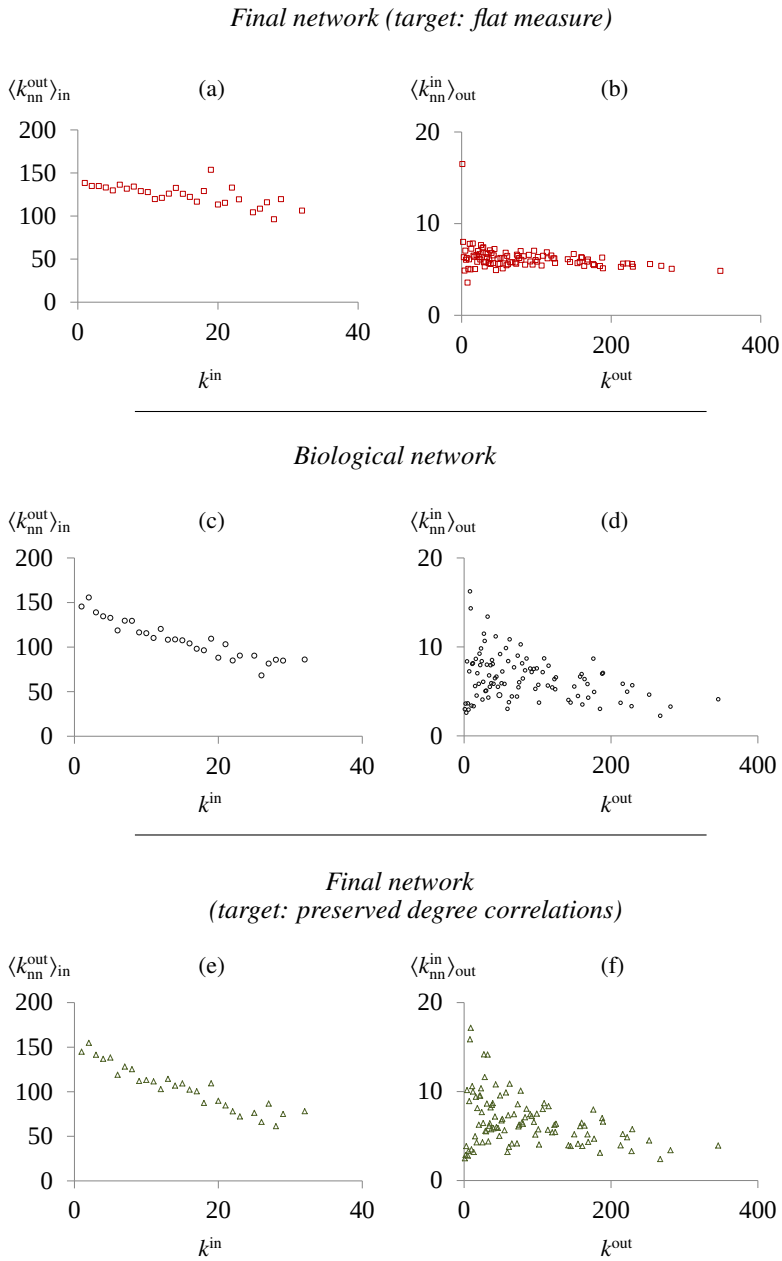


Figure 6.10: These charts summarize the degree-degree correlations observed in the original network (subfigures (c) and (d)), the final network after the process targeting the flat measure (subfigures (a) and (b)) and the process tailored to preserve the degree-degree correlation pattern of the original network (subfigures (e) and (f)). The data used is based on Harbison et al. [2004] and the process shown in figure 6.9. The left hand charts (figures (a), (c) and (e)) summarize the correlation between the in-degree of a node and the average out-degree $\langle k_{nn}^{out} \rangle_{in}$ of its in-neighbours. The right hand charts (figures (b), (d) and (f)) summarize the correlation between the out-degree of a node and the average in-degree $\langle k_{nn}^{in} \rangle_{out}$ of its out-neighbours.

6.5 Switch& Hold method

After the completion of this work, email correspondence with Professor Burda [2012] clarified an alternative approach to unbiased randomisation which is an improvement on the 'Switch & Hold' method (see e.g Miklós and Podani [2004]). The analysis presented above applies for a process which samples from all possible moves (or $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ sub-matrices of the adjacency matrix in the language of Rao et al. [1996]). The classic 'Switch & Hold' method samples from all quadruplets of nodes (i.e. all 2×2 sub-matrices of the adjacency matrix in the language of Rao et al. [1996]). The alternative outlined by Professor Burda was to sample from pairs of edges, whether or not they constituted a valid move. In the language of Rao et al. [1996]), this is sampling from $\begin{pmatrix} 1 & x \\ y & 1 \end{pmatrix}$ sub-matrices of the adjacency matrix. In both 'Switch & Hold' approaches, if an illegal move is selected, then the previous configuration is held. Below we analyse the 'Switch & Hold' approach for the undirected case.

We define a Markov Chain in the usual way:

$$p_{t+1}(\boldsymbol{c}) = \sum_{\boldsymbol{c}'} W(\boldsymbol{c}|\boldsymbol{c}')p_t(\boldsymbol{c}')$$

but in this case we have defined

$$W(\boldsymbol{c}|\boldsymbol{c}') = \begin{cases} 0 & \boldsymbol{c},\ \boldsymbol{c}' \textit{ not neighbours} \\ \frac{1}{T} & \boldsymbol{c},\ \boldsymbol{c}' \textit{ neighbours} \\ 1 - \frac{n(\boldsymbol{c})}{T} & \boldsymbol{c} \equiv \boldsymbol{c}' \end{cases} \tag{6.5.1}$$

where $n(\boldsymbol{c})$ is defined in the usual way. Two states are considered neighbouring if one can be reached from the other with one legal move. T is the number of moves that can act on c. $T = n(\boldsymbol{c})$ corresponds to the usual way we have defined the Markov Chain picking from valid swaps. $T = \frac{N(N-1)(N-2)(N-3)}{4}$ corresponds to the classical 'Switch & Hold' algorithm (see e.g. Miklós and Podani [2004]) - where the sampling is from all possible quadruplets of nodes. If an illegal move is picked then the current state is maintained. Burda [2012] refers to another variant, where the moves are picked from all possible pairs of edges, meaning that $T = 0.5\bar{k}N(0.5\bar{k}N - 1)$. It will be shown that this radically improves the practicality of the 'Switch & Hold' approach.

Ergodicity is satisfied. Let us study the detailed balance for the 'Switch & Hold' algorithm. The crucial feature is that every state has a constant number of possible moves T that can act on it.

Writing

$$W(\boldsymbol{c}|\boldsymbol{c}')p_\infty(\boldsymbol{c}) = W(\boldsymbol{c}'|\boldsymbol{c})p_\infty(\boldsymbol{c}') \tag{6.5.2}$$

This is trivially satisfied for the first and third case: if $\boldsymbol{c}$ and $\boldsymbol{c}'$ are not neighbours then both sides are equal to zero, if they are identical then both side are equal by construction. If $\boldsymbol{c}$ and $\boldsymbol{c}'$ are neighbours, then it follows that $\frac{p_\infty(\boldsymbol{c})}{T} = \frac{p_\infty(\boldsymbol{c}')}{T}$. Hence we have shown that the equilibrium probability distribution is uniform.

This can also be shown diagrammatically by viewing this process as a random walk on a regular graph (where nodes are states, and an edge is drawn where the process can move from one space to another in one step and self-loops indicate failed moves).

If both approaches are unbiased, the discriminator becomes speed and computational complexity. Picking four nodes at random, only very few will meet the condition to form a valid swap moves, so the process would have a tendency to spend a long time in each state. So the process will take a long time to explore the state space, and hence a long runtime to become useful as a means of calculating any kind of ensemble averages or null models.

It is immediately clear that adjusting the 'Switch & Hold' method to sampling from pairs of links will radically decrease the number of failed trials (since typically $\bar{k} << N$). So let us compare this adjusted 'Switch & Hold' sampling (indexed '1' below) with the Markov Chain process based on adjusting the acceptance probabilities of moves proportional to the states mobility (indexed '2' below). The most pertinent comparison will be based on how rapidly the process samples the whole state space. This can be studied by focussing on comparing the rejected moves probabilities for process '1' and '2'.

$$p_1(\boldsymbol{c} \to \boldsymbol{c}) = 1 - \frac{n_{\boldsymbol{c}}}{T} \tag{6.5.3}$$

$$p_2(\boldsymbol{c} \to \boldsymbol{c}) = 1 - \sum_b \frac{n_c}{n_c + n_b}\frac{1}{n_c} \tag{6.5.4}$$

$$\tag{6.5.5}$$

where the sum is taken over all b which are neighbours of c. The final term is the product of the probability of picking a move ($\frac{1}{n_c}$), and the acceptance probability of the move. For an arbitrary

Species	$\frac{\langle n \rangle}{T}$
Cam. jejeuni (Parrish et al. [2007])	0.84
Drosophila melanogaster (Rehwinkel et al. [2006])	0.98
Escherichia coli (Arifuzzaman et al. [2006])	0.88
Hel. pylori (Rain et al. [2001])	0.93
Homo sapiens (Ewing et al. [2007])	0.88
Sac. cerevisiae (Collins et al. [2007a])	0.87

Table 6.1: A variety of biological datasets and their corresponding values for $\frac{\langle n \rangle}{T}$. We would expect 'Switch & Hold' to be faster when $\frac{\langle n \rangle}{T} > 0.5$.

c, which process has the greater probability of staying in the same state?

$$E(p_1(\boldsymbol{c} \to \boldsymbol{c})) = 1 - \frac{\langle n \rangle}{T} \tag{6.5.6}$$

$$E(p_2(\boldsymbol{c} \to \boldsymbol{c})) = 1 - E\left(\sum_b \frac{1}{n_{\boldsymbol{c}} + n_{\mathbf{b}}}\right) \tag{6.5.7}$$

We have actually calculated an expression for the change in mobility. So we know that where N is large and the network is sparse ($\bar{k}$ is order 1) $E(\sum_b \frac{1}{n_{\boldsymbol{c}}+n_{\mathbf{b}}}) \approx \frac{\langle n \rangle}{2\langle n \rangle} = \frac{1}{2}$. So if $\frac{\langle n \rangle}{T} > \frac{1}{2}$ we would expect process 1 to be quicker than process 2. Table 6.1 shows $\frac{\langle n \rangle}{T}$ for some real networks.

So process 1 is expected to explore the space quicker. If we looked at the other end of the spectrum - say a nearly hardcore network with 100 nodes (K=98)- we find that sampling moves from random pairs of edges results in a failed trial 99.9% of the time. Sampling from valid moves means that a move is rejected with 50% of the time. So - for denser networks - choosing between the two methods comes down to a trade off between longer runtimes and more calculation at each step.

The results published are fully correct and accurate (up to ambiguity in some of the cited literature as to precisely how they had chosen to randomize). However, based on the analysis above, it seems that switch and hold sampling from pairs of edges is unbiased and generally faster. Reviewing the literature, it remains ambiguous whether the approaches used were biased or unbiased (see e.g. Bansal et al. [2009], Gotelli and Ulrich [2012], Maslov and Sneppen [2002], Sheppard et al. [2011], Royer et al. [2008], Tabourier et al. [2011]). Unbiased random graph generation is important and not always evident.

6.6 Summary

In this chapter we have built on the work of Rao et al. [1996] and Coolen et al. [2009] to define an ergodic and unbiased stochastic process for randomising directed binary non-self-interacting networks, which keeps the number of in- and out- connections of each node constant. The result takes the form of a canonical Markov Chain Monte Carlo (MCMC) algorithm based on simple directed edge swaps and triangle reversals, with nontrivial move acceptance probabilities that are calculated from the current state of the network only. The acceptance probabilities correct for the entropic bias in 'accept all' edge-swap randomization, which is caused by the state dependence of the number of moves that can be executed (the 'mobility' of a graph).

We have derived bounds to predict for which degree sequences the differences between 'accept all' and correct randomization (i.e. the effects of sampling bias) will be negligible. Application to synthetic networks showed a large discrepancy between the 'accept all' and correct randomization processes, and good agreement with our theoretical predictions for the values of key observables that are affected by the entropic bias of incorrect randomization. For the biological networks which we studied (gene regulation networks) we find the differences between correct and incorrect sampling in the space of graphs with imposed degree sequences to be negligible. However, this cannot be relied upon to continue in future studies, especially when network datasets become less sparse, or randomization processes which target more complicated topological observables are used.

Our process is precise for any network size and network topology, and sufficiently versatile to allow random directed graphs with the correct in- and out-degree sequence to be generated with arbitrary desired sampling probabilities. The algorithm can be used to generate truly unbiased random directed graphs with imposed degrees for hypothesis testing or to generate more sophisticated null-models which inherit from a real network both the degree sequence *and* the degree correlations, but are otherwise random and unbiased.

In chapter 5 we studied random graph ensembles with targeted average numbers of triangles. Simulations for that ensemble were approached with a Metropolis algorithm, where edges are repeatedly proposed to be added or deleted between randomly picked pairs of nodes, and accepted or rejected with a probability driven by the energy function of the current and proposed next state. This is the canonical approach for simulating networks under soft constraints. The

ensembles studied in chapter 4 have hard constraints. We note that the structural relationships observed in that chapter between bipartite ensembles, specified neighbourhood ensembles and generalised degree ensembles also provides a route to define an edge swap randomisation process for these ensembles. For the bipartite ensemble edge swaps apply directly. For the specified neighbourhood case, neighbourhood-preserving edge swaps could be defined on the $\boldsymbol{\beta}$ bipartite sub-networks. For the generalised degree ensemble, the first step could be to randomly select a specified neighbourhood distribution, and then to proceed as previously.

CHAPTER 7

NETWORK ENSEMBLES BASED ON BIOLOGICAL DATASETS

7.1 Applying calculation in chapter 3 to gene regulation networks

A gene regulation network can be viewed as a directed graph, where the nodes represent genes and the arcs indicate whether ($c_{ij} = 1$) or not ($c_{ij} = 0$) the protein synthesized from gene j acts as a regulator of gene i. In the present binary set-up, where $c_{ij} \in \{0, 1\}$, one disregards information on the nature of regulation, i.e. whether it involves repression or activation.

In tables 7.1 and 7.2 we show the results of calculating the various contributions to the entropy of the ensemble associated with the networks of Hughes et al. [2000] and Harbison et al. [2004] respectively [1]. Imposing only the correct average degree gives the entropy $S_0[\bar{k}]$. Imposing in addition the correct degree distribution (i.e. representing the network by ensemble 3.1.1) gives the entropy $S_0[\bar{k}] - C_{\rm deg}[p]$. Imposing additionally the correct degree-degree correlations (i.e. representing the network by ensemble 3.2.1) reduces the entropy still further to $S_0[\bar{k}] - C_{\rm deg}[p] - C_{\rm wir}[p, W]$.

[1] The data was collated from public databases and kindly provided for my use during my visit to the Schlitt group in King's College Genetics Department. Further details about this data can be obtained from Lehne [2012].

In both tables we also show the entropies per arc, defined as $S' = S/\bar{k}$. The latter are normalised for the average degree. This fits in with the 'arc centric' view that the calculations in this chapter and the preceding work Annibale et al. [2009] seem to have steered us in, where the final answers are consistently found to be most elegantly formulated in terms of the joint distribution W of degrees at either end of an arc.

Hughes et al. [2000] used a two-color cDNA micro-array hybridization assay to generate expression profiles in yeast for 276 deletion mutants. We followed an approach published by Rung et al. [2002] to construct a network from this data. Two genes g1, g2 are connected by an arc from g1 to g2 if the ratio of the expression level in the mutant where gene g1 is deleted versus the background standard deviation in the wild-type strain is larger than a threshold. In this way, we arrived at a directed network with $N = 5654$ nodes (genes), with an average degree $\bar{k} \approx 5.6$. The degree distribution of this network is characterised by high frequency of occurrence of low degree nodes; the set of nodes with out-degree zero and in-degree less than 4 covers more than 50% of the set. However, the network also contains some nodes with very high out-degree.

Gene regulation network of Hughes et al. (2000)

Imposed topological property	Entropy per node	Entropy per arc
Average degree	44.5	7.9
Degree distribution	19.5	3.5
Degree-degree correlations	17.9	3.2

Table 7.1: The tailoring of random graph ensembles by imposing as constraints the values of increasingly prescriptive macroscopic topological features measured in the gene regulation network of Hughes et al. [2000]. This tailoring reduces the entropy per node S in the ensemble in stages, and thereby the effective number of graphs $\mathcal{N} = \exp[NS]$ compatible with the network of Hughes et al. [2000]. We observe that, in this example, refining the tailoring of the graph ensemble from imposing only the correct average degree to imposing the correct degree distribution is more significant than the further refinement of imposing the correct degree-degree correlations. Hence the degree complexity of this network is significantly larger than the wiring complexity.

The authors of Harbison et al. [2004], reported on a study of DNA binding transcriptional

regulators in yeast. For each of the 203 transcription factors tested they report the genes where the transcription factor bound to the putative promoter region. Similar to a previous study Schlitt et al. [2003] we constructed a network by connecting gene g1, which encodes a transcription factor, to gene g2 if the measurements were statistically significant ($P \leq 0.001$). Their data were represented as a directed network of $N = 3865$ nodes, with an average degree of $\bar{k} \approx 2.81$. Compared with the data of Hughes et al. [2000], the network of Harbison et al. [2004] is more sparse. It does, however, show a similar degree distribution pattern - in fact over 50% of the nodes have zero out-degree and an in-degree of less than 2.

Gene regulation network of Harbison et al. [2004]

Imposed topological property	Entropy per node	Entropy per arc
Average degree	23.2	8.2
Degree distribution	12.8	4.5
Degree-degree correlations	11.6	4.1

Table 7.2: The tailoring of random graph ensembles by imposing as constraints the values of increasingly prescriptive macroscopic topological features measured in the gene regulation network of Harbison et al. [2004]. The tailoring reduces the entropy per node S in the ensemble in stages, and thereby the effective number of graphs $\mathcal{N} = \exp[NS]$ compatible with the network of Harbison et al. [2004]. As in the previous example, refining the tailoring of the graph ensemble from imposing only the correct average degree to imposing the correct degree distribution is more significant than the further refinement of imposing the correct degree-degree correlations. Hence the degree complexity of this network is again significantly larger than the wiring complexity.

In practice, when the gene network data are collected, a decision has to be made about the cut-off point where the effect of one gene product on another gene is so small as to be considered insignificant. If there was no threshold and every small fluctuation was taken to be evidence of co-regulation, then it would appear that every gene regulated every other gene, and the network would be complete. Conversely, setting too strict a threshold will risk missing out on important but subtle interactions.

Changing the threshold would reduce the number of arcs, and hence make the network more

sparse with lower average degree. Our base assumption would be that beyond that, the main qualitative features of the topology would be maintained. That is, the stricter threshold would remove arcs indiscriminately across the network. However, it is possible that, for example, a node would appear to be a 'hub' under a lenient criterion, but would lose a large number of interactions under stricter criteria, so that it is no longer a hub: this would be a qualitative change to the topology arising from the change in thresholds. The analysis proposed in this chapter is measuring the topological properties of the network (rather than the network itself). We would expect these results to vary insofar as the topological properties varied. Figure 7.1 shows the results of repeating the analysis above for different values of the thresholds.

The above data all refer to the same organism, yeast; however, they present different aspects of gene interactions. Hence, even more than for protein-protein interaction networks, comparison must be done cautiously. The heterogeneity in the data sets emphasises the importance of developing a suite of tools and measures that can be used to study each network independently.

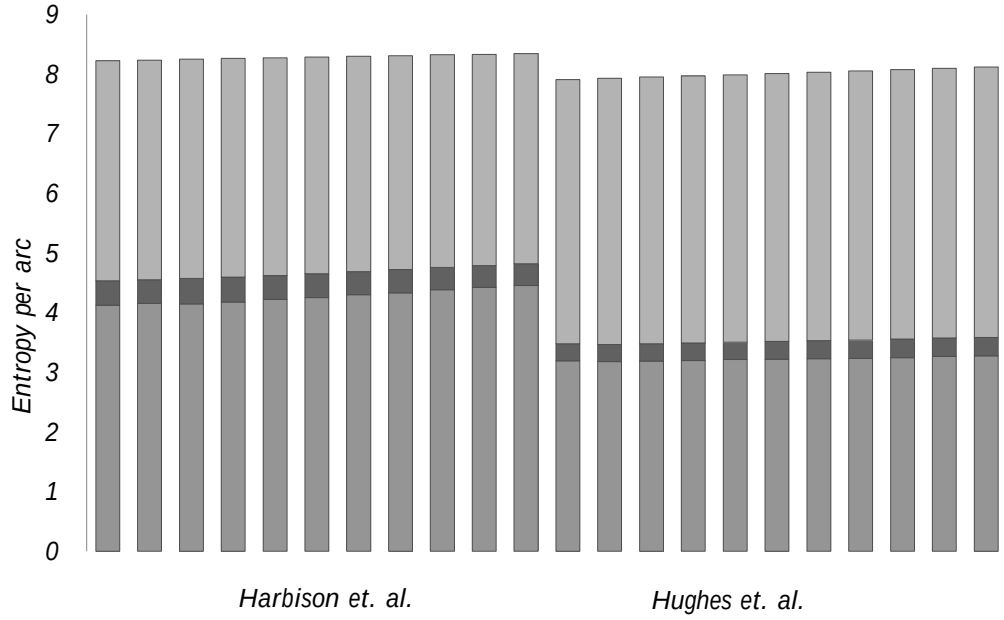


Figure 7.1: Each bar on the chart represents a different choice of threshold. Moving from left to right, the threshold is made progressively stricter so as to exclude approximately 3 percent of arcs at each step. Within a bar, the top line presents the entropy per bond when the constraint is 'average degree'; the next line shows the entropy per bond when the constraint is additionally 'degree distribution'; and, the final line gives the entropy per bond for the ensemble additionally targeting the 'degree-degree correlation'. Hence the top two shaded areas represent the degree complexity and the wiring complexity respectively. Both datasets are plotted on the same axis in order to illustrate that, although there is some movement with different thresholds, the results for the two different networks remain distinct and distinguishable for any reasonable choice of threshold, and are not unduly sensitive to any reasonable choice of threshold.

7.2 Illustrations of concepts and results relating to generalised degrees (Chapter 4)

Specifying the generalised degree distribution is an onerous restriction on a random graph ensemble. As an illustration, Figure 7.3 shows this numerically by applying equation 4.5.10 to random graph ensembles tailored to the topology of real biological networks, whose generalised degrees are shown in Figure 7.2. The results show that, generally, incorporating the generalised degree constraint in the definition of the ensemble results in substantial reduction in the associated Shannon entropy. The magnitude of this reduction correlates well with a qualitative inspection of the heatmaps of the associated generalised degrees. Specifically, comparing qualitatively confirms that there is greater complexity due to generalised degree where the generalised degree shows greater divergence from the maximum entropy simple degree distribution. Figure 7.4 combines Figure 7.2 with Figure 7.3 to allow for direct visual comparison of the qualitative features of the generalised degree distribution obtained from these datasets with the quantitative value for the complexity associated with the same distribution.

Figure 7.5 plots the generalised degree heatmaps of some simple synthetic networks in order to demonstrate how generalised degrees are a much more specific topological signature than simple degrees. In the first network - a simple chain - every node has the same generalised degree: $(2, 4)$. The generalised degree heatmap has a single peak. The second network - a star - has generalised degree distribution $p(k, m) = \delta_{(k,m),(99,99)} \frac{1}{100} + \delta_{(k,m),(1,99)} \frac{99}{100}$. There is a visible peak at $(1, 99)$. This peak is deformed due to smoothing with the peak at $(99, 99)$ (which is too faint to see in its own right). The final network is a Cayley tree. Every interior node has generalised degree $(3, 9)$. View the Cayley tree as being grown one layer at a time (i.e. at every layer attach two new nodes to each node in the preceding layer), for n steps. The final two layers will have irregular generalised degree values (since they lack any further outwards neighbours) - being $(1, 3)$ and $(3, 6)$ for the final and penultimate layer respectively. Their relative frequencies can be calculated by summing a geometric series, and correspond to the three peaks shown.

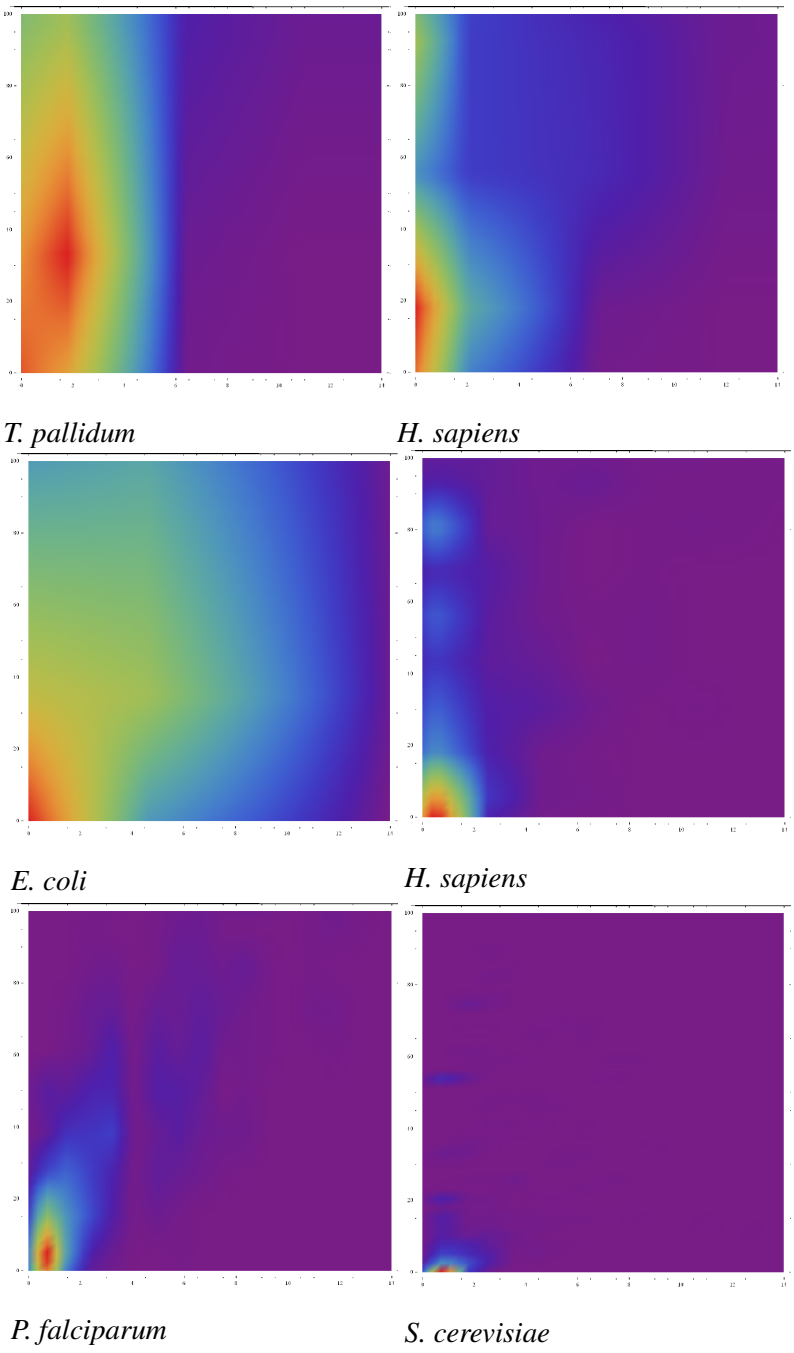


Figure 7.2: Generalised degree distributions in real biological networks. These plots are cross-referenced with the quantitative results in figure 7.4. The heatmaps are smoothed; the x axis corresponds to the first degree and the y axis corresponds to the second degree; the intensity of the colour indicates the observed frequency of that pair of values.

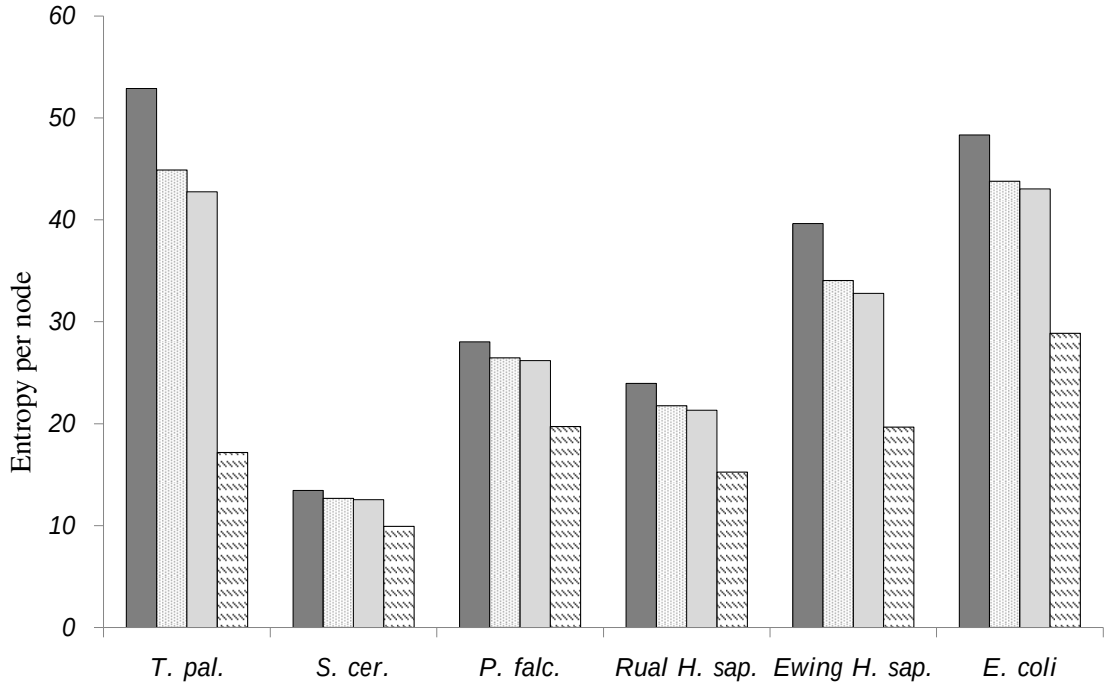


Figure 7.3: Results from applying equation 4.5.10 for the entropy of random graph ensembles - where the constraints are taken to match the relevant topological observables of networks from Titz et al. [2008], Ito et al. [2001], LaCount et al. [2005], Rual et al. [2005], Ewing et al. [2007], Arifuzzaman et al. [2006]. From left to right the bars correspond to entropy per node of random graph ensembles tailored to match: average degree, degree distribution, degree-degree correlation and generalised degrees. For each dataset, the difference between two bars quantifies how much more restrictive the subsequent constraint is. It is clear that generalised degrees are typically a very demanding constraint, whereas a specified degree-degree correlation does not add a lot of information beyond the degree distribution.

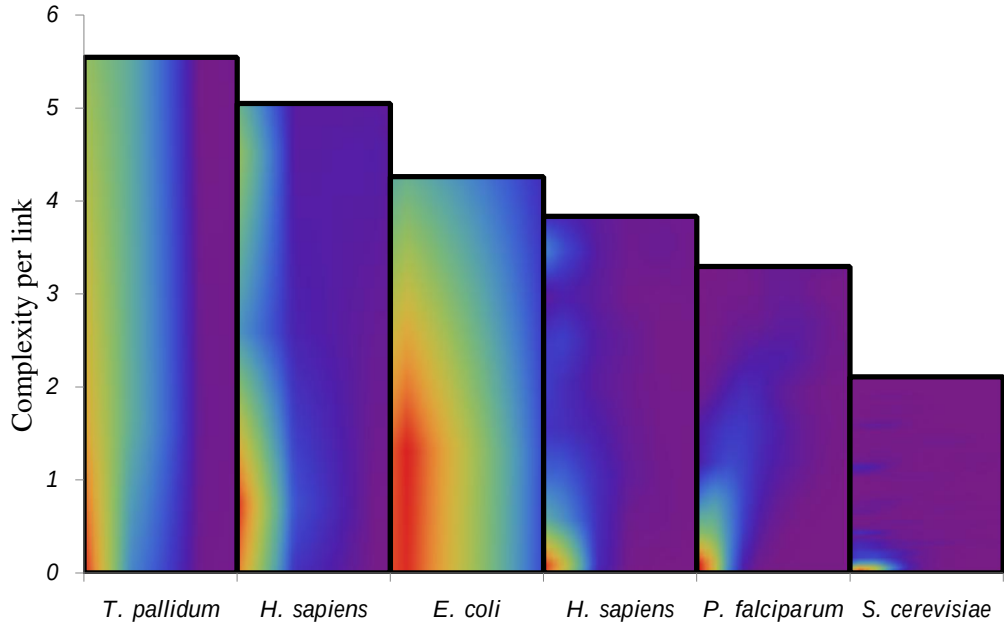


Figure 7.4: This plot compares the calculated value for the complexity associated with the generalised degree constraint with the appearance of the generalised degree heatmap plot. From left to right along the x-axis the bars correspond to data from Titz et al. [2008], Ewing et al. [2007], Arifuzzaman et al. [2006], Rual et al. [2005], LaCount et al. [2005], Ito et al. [2001]. The y-axis gives the complexity per bond associated with the 'generalised degrees' constraint (as taken from the corresponding dataset). This data is equivalent to subtracting the level of the fourth bar from the level of the second bar in figure 7.3 and then normalising by dividing by the average degree $\bar{k}$. The bars are reordered from largest to smallest for easier comparison. By inspection, it can be seen that there is a trend in the images which corresponds to intuitive expectation. The heatmaps are smoothed; the x axis corresponds to the first degree and the y axis corresponds to the second degree; the intensity of the colour indicates the likelihood of that pair of values.

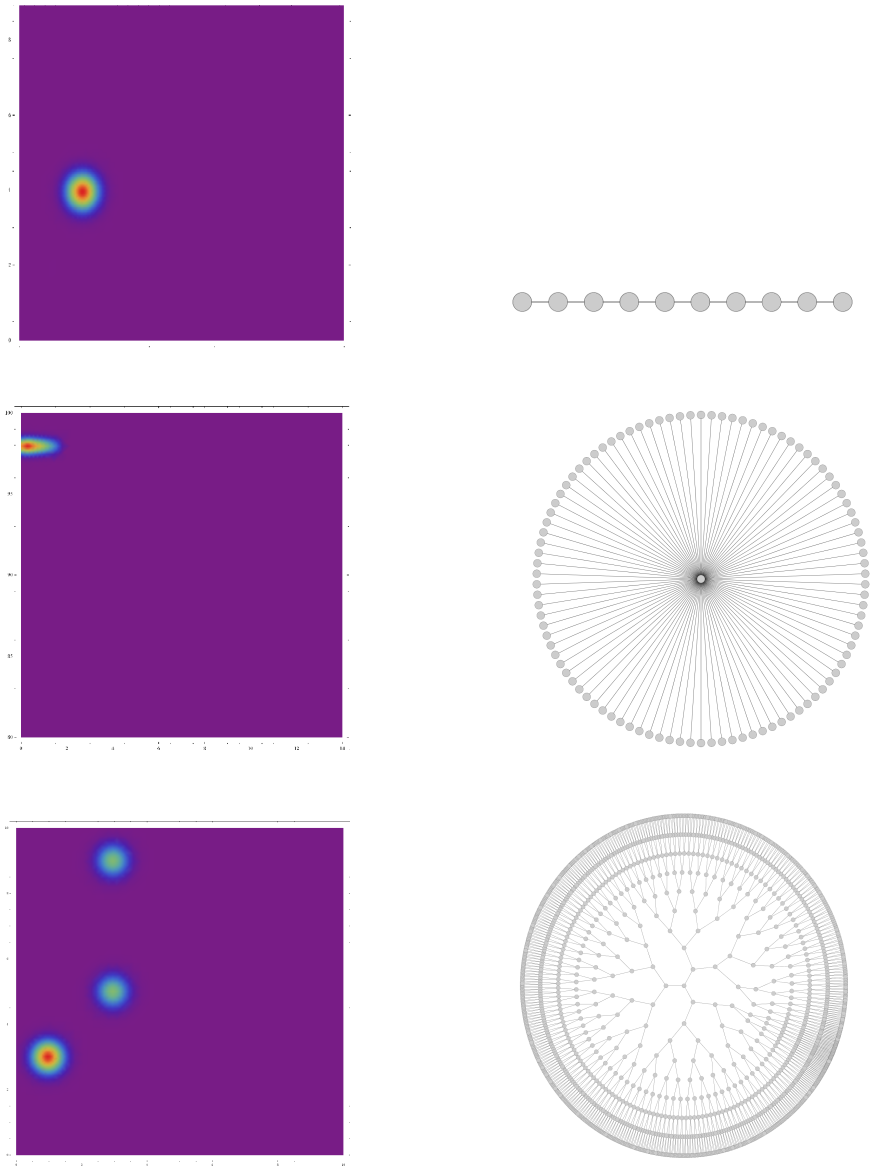


Figure 7.5: Some simple networks and their generalised degree heatmap. The first degree is on the x axis and the second degree is on the y axis. The colour at any co-ordinate indicates the frequency of occurrence of that pair of values. The plots are smoothed.

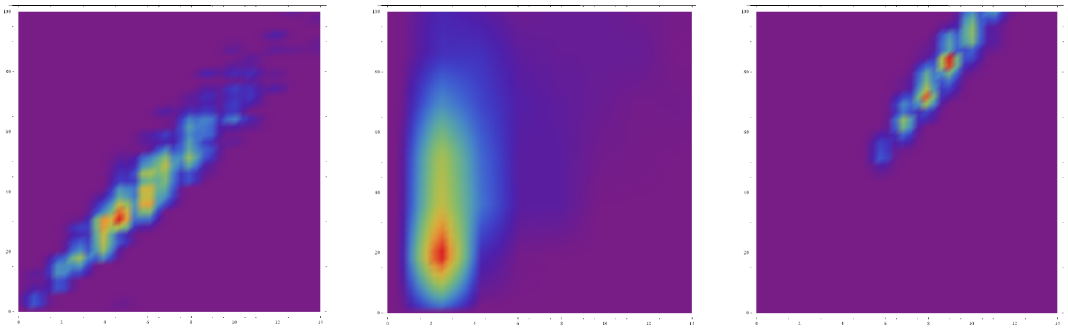


Figure 7.6: Example generalised degree distributions. From left to right: an Erdös and Rényi [1960] type network with links appearing with constant probability; a network generated with the Barabási and Albert [1999] preferential attachment model; and, a small-world type random network with enhanced clustering, based on the algorithm by Watts and Strogatz [1998]. The heatmaps are smoothed; the x axis corresponds to the first degree and the y axis corresponds to the second degree; the intensity of the colour indicates the likelihood of that pair of values.

CHAPTER 8

CHOICE OF NULL MODEL FOR NUMERICAL INVESTIGATIONS INTO MOTIF ABUNDANCE

In this chapter we carry out two illustrative pieces of analysis in order to investigate how the choice of null-model could influence the conclusions of statistical analyses. In particular, we focus on several examples of a common bioinformatics strategy: the identification of statistically over-represented motifs in a network. By using a wider range of null-models, we can provide greater challenge to whether the observed feature is exceptional enough to need further investigation. It can also give clues where the motifs may be a by-product of another topological property.

The purpose of doing this analysis was to promote using a variety of null-models to base any claims of a motif being statistically over-represented. We add one new null model: the one with controlled degree-degree correlation. However - as variously outlined in this thesis - there is a wide range of constrained network ensembles which can be meaningfully defined and numerically generated. The degree constrained null-model is well established, but this is primarily due

to the easy availability of software and algorithms to generate examples from this ensemble. To the best of our knowledge, there is no conceptual reason to expect this to be a gold-standard null-model.

The randomisation approach is based on Coolen et al. [2009] - albeit the computer code was re-written especially for this analysis.

8.1 Power graph analysis to estimate motif density

Royer et al. [2008] published a paper which presented a method of compressing a graph into a so-called 'power graph', where certain motifs (stars, cliques and bicliques) were represented by special nodes. This paper was used as a starting point primarily because the associated software gave an easy and immediate way to measure the density of motifs in real and randomised networks. Hence the objective is not to critique Royer et al. [2008], but rather to provide a demonstration of the strong interdependence between motifs and degree-degree correlations.

Royer et al. [2008] validated their results with reference to a degree constrained model. Our analysis looks at the real network, the degree constrained null-model, and a second null-model which was constrained to share both the average degree and the degree-degree correlation of the original network. Figure 8.1 presents the results in the same format as figure 5 of Royer et al. [2008]; including the extra null-model does not dispute their overall finding that real networks were significantly richer in cliques than would be expected to occur randomly. Table 8.1 shows that the significance of an observation varies significantly with the choice of null models, and there is no clear pattern to the scale or direction of the variation. Figure 8.2 is a geometric representation of this data.

Dataset	Royer et al. [2008] Z value	Calculated Z value degree-controlled null model	Calculated Z value degree-degree correlation null-model
Rual et al. [2005]	7.3	6.61	12.11
Ewing et al. [2007]	43.2	47.0	27.5
LaCount et al. [2005]	2.2	3.3	0.03
Arifuzzaman et al. [2006]	13.3	13.7	19.6

Table 8.1: This table shows the Z score of the observed motif density, versus the distribution of motif densities in networks from tailored random graph ensembles. The Z score measures how many standard deviations away from the mean of the distribution the observation falls. As our analysis is 'proof of concept' only, we satisfied ourselves at looking at 4 datasets, with 15 instances generated for every option. Royer et al. [2008] generated 1000 instances for every option, hence columns 2 and 3 give the scale of the 'small sample' effect. Column 4 provides the new result: Z values for the degree-degree correlation null model.

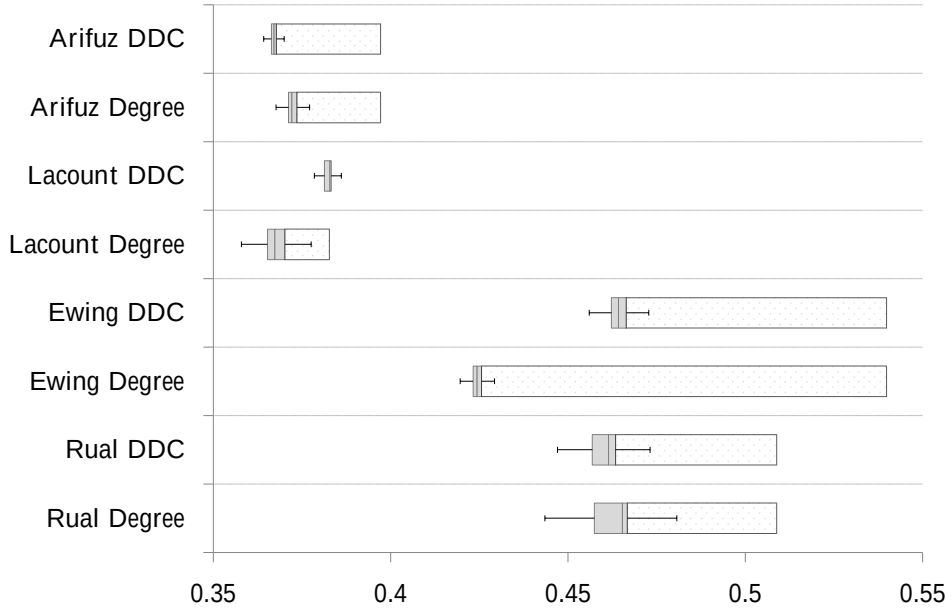
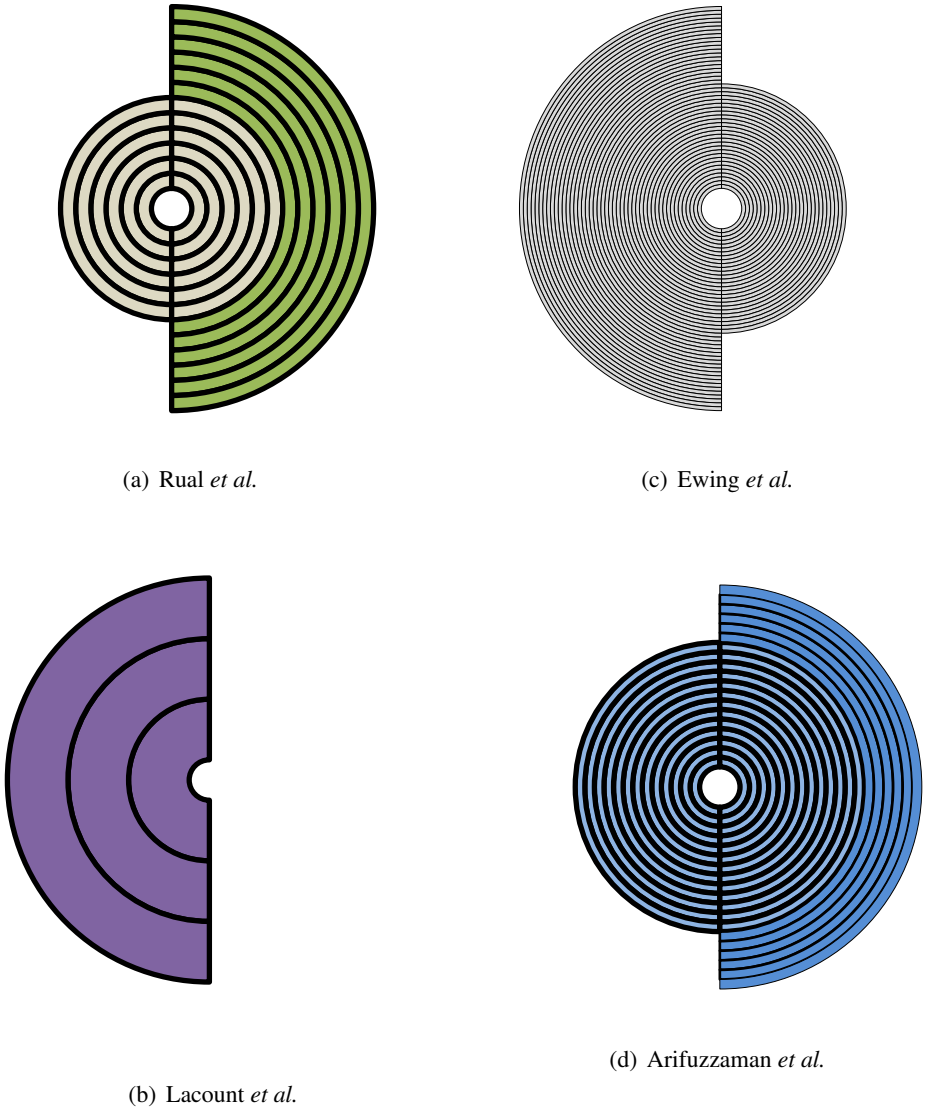


Figure 8.1: A repeat of the analysis from Royer et al. [2008] on the 'compressibility' of real biological protein interaction networks (indicated on the x-axis). Compressibility is related to the number of star, clique and bi-clique motifs in the network. Similar to Royer et al. [2008], we analysed the compressibility of the real network (dotted bars) and the degree-constrained randomised network (box and whisker plots labelled 'degree'). The box contains 50% of observed values). The novelty is the addition of another null model (DDC) which shares the degree distribution and the degree-degree correlation of the original model. Dataset references are the same as table 8.1. The dotted region extends from the mean of the null model to the actual observation.



(a) Rual *et al.*

(c) Ewing *et al.*

(b) Lacount *et al.*

(d) Arifuzzaman *et al.*

Figure 8.2: An alternative way of displaying the data from figure 8.1. Closeness of observed motif density to the mean found in the degree-controlled ensemble (left hemisphere) and the degree-degree correlation controlled ensemble (right hemisphere). Every ring indicates one standard deviation. This chart is provided to illustrate how multiple null-models can be aggregated into a single geometric chart, to give a quick visual representation of the wider context of the observation; the circle could be divided to accommodate any number of null models.

8.2 A network of Olympics 2012 events, and its null models.

In this section we construct a network based on events at the London Olympics in 2012. The purpose is to provide an example — which is idiosyncratic but real – where the degree constrained null model and the degree correlation constrained null model diverge substantially. In this network, nodes are events, and two nodes are linked if at least one competitor entered both events. At the London 2012 Olympics, Michael Phelps won four gold medals and two silver medals. This, together with his previous victories, led to some commentators calling him the greatest ever Olympian [1]. Critics observed that, as a swimmer, he had the opportunity to enter more events than other athletes. In this section, we will use our network of Olympic events, and various null models, in order to provide some statistical evidence for whether the structure of the current Olympic games is unusually likely to create a super-Olympian. The reasoning is that for one individual to win a large number of events, they must have the opportunity to enter a large number of events: the largest possible number of medals that can be won is the same as the largest clique in the network. Six medals is indeed a remarkable achievement; the analysis looks for the number of opportunities to win at least three medals, by equating this with the number of triangles in the network. The analysis is repeated for four different randomised networks. The first null-model was an Erdös-Rényi model with the same average degree. The second null model used accept-all edge swapping to generate a null-model with the same degree distribution (but this is not be a representative model from this ensemble, since accept-all edge swapping is known to be biased). The third null-model also preserved the degree distribution, but achieved this via the unbiased MCMC algorithm from Coolen et al. [2009]. The final null model used the algorithm from Coolen et al. [2009] to generate a null-model with the same degree-degree correlation. It is evident from the figures that the actual organisation of Olympics events gives many more opportunities to win three or more medals, compared to null-models with appropriately controlled topological parameters. A visual inspection of figure 8.3 shows that the network has two dense cores, corresponding to male and female swimming disciplines. This can also explain why there is such a significant deviation in performance between the biased and unbiased methods of generating degree-controlled null models.

[1] http://www.theguardian.com/ sport/blog/2012/aug/04/london-2012-michael-phelps-olympian

Null-model	Number of triangles
Controlled average degree	7
Controlled degree distribution (naïve accept all swaps version)	15.6
Controlled degree distribution (unbiased version)	36.8
Controlled degree-degree correlation	69.4
Real network	247

Table 8.2: Number of triangles found in the Olympic events network, and in a selection of tailored null-models. The numbers have not been normalised. For context, the data set was 10,384 athletes competing in 300 events.

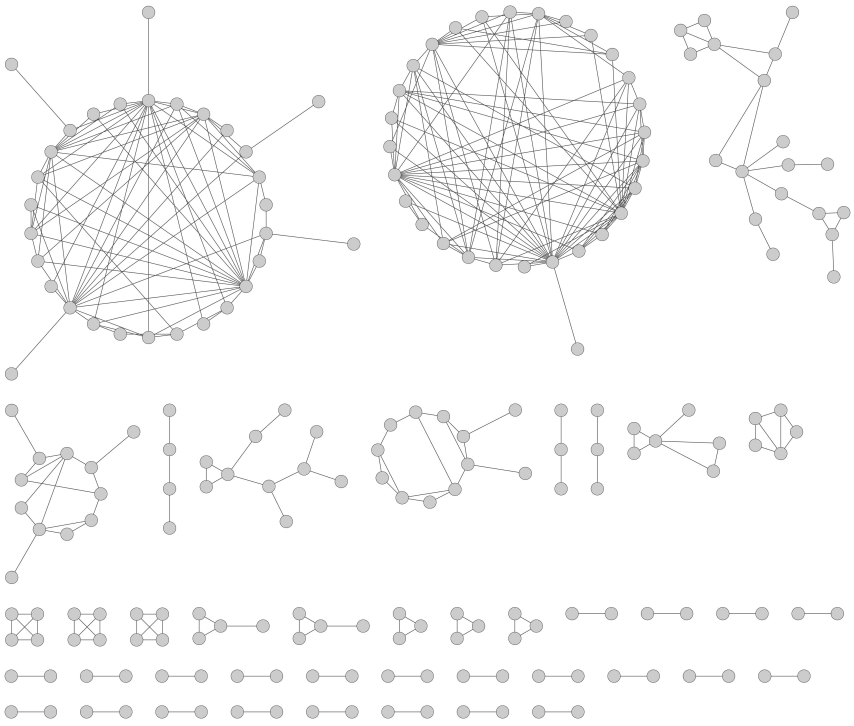


Figure 8.3: Olympics 2012 network of complementary Olympic events. Nodes are the medal events. Two nodes are linked if the corresponding two events had at least one competitor in common at the London 2012 Olympic Games. Isolated nodes excluded. Data visualised with Cytoscape and deduced from athlete data from www.guardian.co.uk/data .

CHAPTER 9

A STUDY OF TOPOLOGY AS A PREDICTOR OF LETHALITY

9.1 Introduction

This chapter will develop a method of ranking nodes based on the formulae in Annibale et al. [2009], apply this to *Sac. cerevisiae* protein-protein interaction networks and compare the resulting ranking with lethality studies. A protein is considered lethal if its removal results in the organism being non-viable. Additionally, in order to build intuition, the method is applied to the network of airline route maps. The motivation for pursuing this line of research is neatly summarised by Acencio and Lemke [2009]

> "The identification of essential genes is important for the understanding of the minimal requirements for cellular life and for practical purposes, such as drug design. However, the experimental techniques for essential genes discovery are labor-intensive and time-consuming. Considering these experimental constraints, a computational approach capable of accurately predicting essential genes would be of great value."

Manke et al. [2006] performed a similar study, which used their own definition of network entropy, and found that the probability that a node is lethal correlates with its entropic contribution.

This idea is as follows: if we remove a single node from the network, the degree distribution and the degree-degree correlation will be perturbed slightly. We can then recalculate the entropy of the constrained random graph ensemble. This will be related to the extent to which this node is perturbing the distribution away or towards the random baseline. This approach is used to generate a ranking for nodes within a network based on their impact on the complexity contributed by specified topological properties.

As an illustrative application, the rankings achieved with this method are compared with *Sac. cerevisiae* lethality data. There is a long running debate as to whether topology is predictive of lethality. Jeong et al. [2001] introduced the idea of lethality-centrality, which suggested that highly connected nodes are more likely to be lethal due to their central role within the topology of the network. Subsequent authors (e.g. Coulomb et al. [2005]) have suggested that observed correlations may be entirely attributable to network artefacts. The success that can be expected from predicting phenotypes via topology must lie somewhere in between. Networks are ultra-simplified representations of a cell, and lethality data has many well documented nuances and drawbacks (e.g. Acencio and Lemke [2009]). Hence, it is clear that no topological tool could be fully predictive of lethality. However, the extent of ongoing research in Systems Biology reflects a consensus that aggregating available experimental knowledge in networks provides a unique opportunity for insight into more subtle and complicated processes which define how the organism functions as a whole.

Ma and Mondragón [2012] focussed on nodes located on community boundaries, noting that these nodes were capable of fragmenting a network. del Rio et al. [2009] find dramatically improved performance when predictions are based on combining two or more measures of centrality. While heuristics like closeness, betweenness and clustering index (such as used by Hwang et al. [2009]) help give us an insight into the qualitative properties of the network, as soon as we start comparing them quantitatively we impose stated and unstated assumptions. For example, a similar property can be defined through a quantity that grows linearly or exponentially with respect to some other quantity. Particularly when measures are combined, inferences may become biased and unreliable.

The proposed new method measures the relative topological significance of each node. It takes into account the degree of the node, but also discriminates within nodes of the same degree using degree-degree correlation. As our suite of Tailored Graph Ensemble (TGE) formulae grow, this idea will have greater impact as an effective method of integrating and comparing topological criteria. For example, Van Kerrebroeck and Marinari [2008] suggest that importance is related to the number of closed paths passing through a node. We could contribute to such a debate once the TGE formulae suite includes the loops constraint (see the material in chapter 5). Hence, the TGE method is potentially a significant step forward compared to heuristic methods of quantifying topological features.

This chapter will test this approach against *Sac. cerevisiae* lethality data. This premise requires some large conceptual steps and connections. We associate reduction in the entropy of the specified ensemble with damage to the topological design of the network, which is assumed to lead to the network being frustrated and prevented from functioning in its normal way. If the network is not functioning properly, then this may be observed experimentally as a reduction in the fitness of the organism. This provides the link to yeast gene lethality studies, although these are known to be incomplete and need to be used advisedly.

9.2 Literature review

Lethality data cannot be fully relied upon. It is not complete, and it requires interpretation. Nonetheless it is an attractive area for research, holding out promise to, for example, improve the success rate of expensive drugs trials. Several authors have proposed that the topological properties of proteins in interaction networks could be strongly related to gene essentiality and cell robustness against mutations (e.g Jeong et al. [2001], Maslov and Sneppen [2002], Barabasi and Oltvai [2004]). Jeong et al. [2001] was an early contributor to the debate, publishing the following encouraging results

> "We rank-ordered all interacting proteins based on the number of links they have, and correlated this with the phenotypic effect of their individual removal from the yeast proteome. ... the likelihood that removal of a protein will prove lethal correlates with the number of interactions the protein has. For example, although proteins with five or fewer links constitute about 93% of the total number of proteins,

> we find that only about 21% of them are essential. By contrast, only some 0.7% of the yeast proteins with known phenotypic profiles have more than 15 links, but single deletion of 62% or so of these proves lethal. This implies that highly connected proteins with a central role in the network's architecture are three times more likely to be essential than proteins with only a small number of links to other proteins."

More recently, Hwang et al. [2009] systematically investigated the topological role of lethal and non-lethal genes in PPI networks, and found that many topological features were statistically discriminative between essential and non-essential genes. Features studied and found significant included: degree, betweeness centrality, closeness centrality, clustering coefficient, neighbours intra-degree and clique level.

While lethality-centrality is an appealing concept, authors such as Palumbo et al. [2005] had a different intuition regarding what makes a node essential. They suggested that in highly connected portions of the network, the availability of alternative pathways makes it possible for an organism to recover from the elimination of any single node. In their view, candidates for lethality should be sought at the periphery of networks and in positions where they connect separate regions of the graph.

There have been several papers which use machine learning to attempt to identify the topological properties which are most predictive of lethality. For example, Acencio and Lemke [2009] proposed a machine learning algorithm relying on network topological features, cellular localization and biological process information for prediction of essential genes, which culminated in a 'decision tree' to predict gene essentiality. This is a brute force method which heavily relies on the quality and abundance of Gene Ontology annotations, which introduces an immediate bias: known essential genes are likely to have fuller annotations. Coulomb et al. [2005] observed that essential genes have on average 175% more SGD curated references than non-essential genes. This may in itself explain any correlation between degree and essentiality.

9.3 Method

For each network of N nodes, N extra networks were generated by removing one node in each case, as well as all the links that the node participated in. Then the degree-degree correlation of the perturbed network is quantified by calculating the entropy of the associated random graph

Name of dataset	Number of proteins overall	Kendall τ overall	Number of proteins with degree $\in (5, 20]$	Kendall τ on deg. $\in (5, 20]$
Krogan et al. [2006]	2708	0.749	649	0.524
Ito et al. [2001]	787	0.741	27	0.330
Tarassov et al. [2008]	1078	0.686	189	0.314

Table 9.1: Kendall's τ is a rank correlation test which takes values between -1 and 1, where 0 indicates that two ordinal rankings are uncorrelated, and +1 and -1 indicate a perfect alignment in rankings, or reversed rankings respectively. It is used here in order to assess whether ordering nodes based on their contribution to the degree and wiring complexity is materially different to ordering the nodes based on their degrees only.

ensemble. The ranking of the removed node is based on this number. This is the *TGE ranking*. The *degree ranking* is simply based on degree. Table 9.1 demonstrates that the TGE ranking and the degree ranking are typically substantially different, particularly when considered within the subset of *sub-hub* nodes which have degrees between 5 and 20. These rankings are cross referenced with lethality studies to construct ROC curves and calculate Gini coefficients. The ROC curve plots the true positive rate against the false positive rate. Gini is defined as twice the area under the ROC curve, minus 1, expressed as a percentage. A result of 0% indicates 'no better than random'.

The results were analysed in the full network, and in the sub-hub region where the degree lies between 5 and 20. Greater than 5 links is the definition of hub used by Han et al. [2004] - but more generally true hubs are considered to be 20 links or above. Comparing rankings is not very meaningful for either very low degree nodes or very high degree nodes. At very high degree nodes, the degree complexity is dominant, and both measures rank nodes identically. At very low degree node, the degree is not discriminating between nodes. The nodes in the sub-hub region may be particularly interesting for, for example, drugs targets, where you would wish to fatally disrupt a disease network, without causing too much damage to the overall network of the disease host. Hence, hubs may need to be immediately discounted, and it may be necessary to be able to discriminate between the lower degree nodes.

9.4 How does the performance of the two rankings compare

ROC curves were used to compare the rankings' effectiveness at identifying lethal and non-lethal nodes for these datasets plus three additional *Sac. cerevisiae* datasets (Collins et al. [2007b], Uetz et al. [2000], Von Mering et al. [2002]). These are not inverses of one another, as lethality studies are not comprehensive. By the lethality-centrality postulate, for the non-lethal case we would expect a negative Gini coefficients. For the lethal case we would intuitively expect positive Gini coefficients. This would indicate that a node with higher degree or greater contribution to the entropy is more likely to be lethal, and less likely to be non-lethal. The first

	LETHALS		NON LETHALS	
	Full Data	Sub hubs	Full Data	Sub hubs
Old (degree) metric	7.5%	-4.5%	-10.9%	-1.5%
New (TGE) metric	6.6%	8.6%	-9.5%	-9.4%

Table 9.2: Gini coefficients averaged over all the datasets studied

notable thing is that both of our topology-driven rankings perform poorly at predicting lethality or nonlethality. The TGE ranking's performance is a substantial improvement on the degree ranking's performance when restricted to the sub-hub region.

Figure 9.1 shows sub-hub lethality results on individual data sets to show that, on the sub-hub region, in most cases the net metric which incorporates degree degree correlation performs better than a ranking based on degree only.

9.5 Airlines

Biological networks are complicated and full of nuances. In order to build intuition on the method, it was applied to a network of airline routes [1]. Again it was observed that we got substantially different rankings with the TGE ranking and the degree ranking. Figure 9.2 shows nodes which appear in only one top 100 list in order to highlight the difference between the two rankings. The difference can almost be entirely attributed by the new ranking featuring US

[1] Data from *openflights.org* .

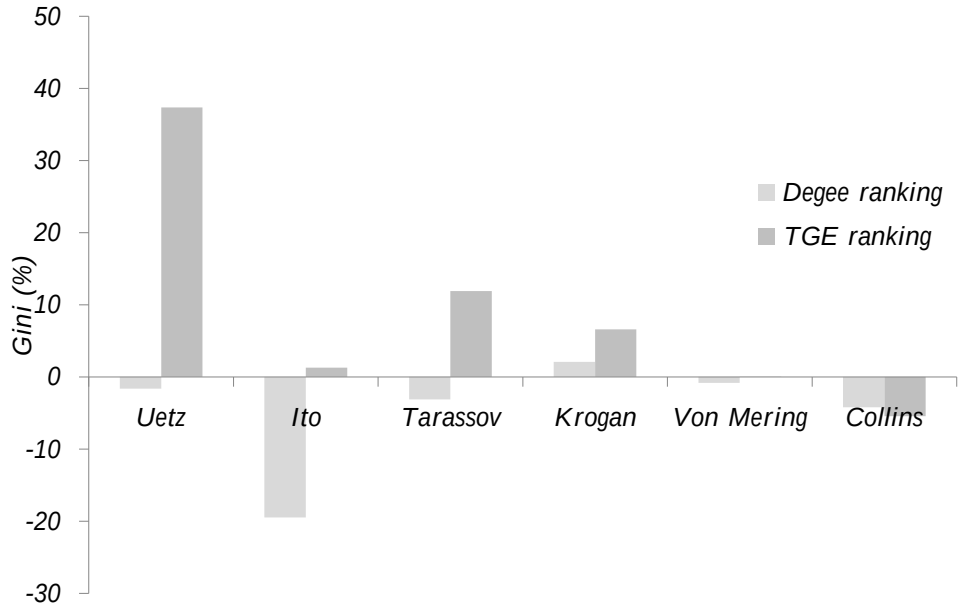


Figure 9.1: Effectiveness at identifying lethal nodes in sub-hub region. The *y* axis measures the Gini coefficient. A highly positive result indicates that the ranking is strongly correlated with lethality.

Figure 9.2: Comparing the location of the top 100 airports under the two rankings. Top map gives the locations of aiports which only appear top 100 in the TGE ranking. Bottom map gives the locations of aiports which only appear top 100 in the degree ranking. By visual inspection it can be seen that the TGE ranking favours the dense USA regional air network, which the degree ranking favours the European international airports. Is there an analogy with cellular processes?

regional airports in favour of European international airports. So, by picking out nodes which most influence the degree-degree correlation of the network we identify a complex *region* of the network - being the dense US domestic airline sub-network. The degree only metric is insensitive to the context of the node in the network, and simply picks out the most highly connected ones.

Considering this, the next step of the work would naturally be to cross reference the biological rankings with the localization of the proteins involved. The aim would be to identify whether - in a similar way to which our ranking favoured certain geographical regions - we can associate the topological properties with functional enrichment. It is interesting to note that Zotenko et al. [2008] propose that proteins are essential due to their involvement in densely connected clusters of proteins with same GO term annotation. Preliminary inspection of the proteins highly ranked by the TGE ranking suggested that there may be material enrichment within organelle membranes.

Table 9.3: Gini coeffiecients showing the predictive performance of ranking nodes based on degree only.

Data source		LETHALS	NON LETHALS	
	Full Data	Sub hubs	Full Data	Sub hubs
Uetz et al. [2000]	1.09%	-1.62%	-1.04%	2.21%
Krogan et al. [2006]	12.25%	2.07%	-25.98%	-22.65%
Ito et al. [2001]	14.59%	-19.47%	-20.64%	12.92%
Tarassov et al. [2008]	-0.25%	-3.10%	-1.22%	2.66%
Von Mering et al. [2002]	13.66%	-0.82%	-9.60%	2.39%
Collins et al. [2007a]	4.00%	-4.22 %	-6.97 %	5.06 %

9.6 Conclusions

The proposed new metric was generally more effective at predicting lethality within the sub-hub region, where the method was expected to be more relevant, commensurate with what we know about the significance of the 'degree-degree correlation' property, and the limitations of predicting lethality with topology.

Table 9.4: Gini coefficients showing the predictive performance of ranking nodes based on their contribution to the degree-degree correlation entropy.

		LETHALS	NON LETHALS	
	Full Data	Sub hubs	Full Data	Sub hubs
Uetz et al. [2000]	0.79%	37.35%	2.12%	-26.49%
Krogan et al. [2006]	8.37%	6.61%	-25.25%	-18.54%
Ito et al. [2001]	16.10%	1.27%	-19.53%	6.83%
Tarassov et al. [2008]	-0.07%	11.90%	-4.55%	-10.53%
Von Mering et al. [2002]	13.71%	0.06%	-7.47%	-0.06%
Collins et al. [2007a]	0.93%	-5.44%	-1.99%	11.36%

We do not find strong evidence that topology is a good indicator of lethality. In particular, within the sub-hub region of interest, the degree metric was shown to be ineffective at predicting lethality. The TGE metric performed better. The difference is small, but this is to be expected. Firstly, there are well documented drawbacks to predicting lethality through topology. Secondly, we are using only one additional topological property at this time. Overall, this shows that tailored graphs methods may give a basis to attach a rational information-theoretic topological metric to individual nodes as well as to entire networks. Testing the method on the simpler airlines dataset suggests that the most interesting direction for further work might be to look at the localization of proteins, compared to how highly they are ranked under each scheme.

CHAPTER 10

DISCUSSION AND CONCLUSION

This thesis has set out to extend the tools available to study ensembles of random graphs which are constrained via certain topological properties. Ensembles of tailored random graphs are extremely useful constructions in the modelling of complex interacting particle systems in biology, physics, computer science, economics and the social sciences. They allow us to quantify topological features of such systems and reason rationally about their complexity, as well as to define and generate useful random proxies for realistic networks. A particular benefit of the approach followed in this thesis is the explicit nature of the final formulae. Although the derivations are involved in places, the final results are compact. They take easily measured topological observables as input, avoid the need for numerical simulations or approximations, and are easy and efficient to use as the (biological) datasets grow. One can therefore imagine that this line of research will continue to develop, by adding further macroscopic network observables, with each addition making the method more powerful and flexible. The core results of the thesis are set out in the paragraphs below.

In chapter 3 we succeeded in rigorously deriving exact closed form solutions for the leading order Shannon entropy of random graph ensembles constrained by directed degree distribution and directed degree-degree correlation. In chapter 4 we calculated the Shannon entropy of ensembles constrained with bipartite degree distributions, specified neighbourhoods and gen-

eralised degrees. The results on directed networks were derived by extending Annibale et al. [2009] to formulate the problem in a path integral form that can be evaluated by steepest descent. A corollary (which could in principle be extended to the subsequent results) is the derivation of an expression for the symmetrised Kullback-Leibler distance between two such ensembles. We derived the Shannon entropy of random graph ensembles constrained by generalised degrees (number of neighbours that can be reached within one and two steps) by resolving the self-consistency expression for the unknown parameter γ derived by previous authors. This was achieved by formulating the implied degree-degree correlation of the ensemble in terms of γ. Placing a natural physical interpretation onto the terms of the expression allows the parameter γ to be eliminated. By designing some straightforward bijective mappings, existing results are leveraged to calculate the entropy of ensembles of random degree-constrained bipartite graphs, and graphs with specified local neighbourhood distributions. Overall, this substantially increases the range of topological properties for which there are exact and explicit results available for the entropy of the associated random graph ensemble.

Chapter 5 reviews some approaches to include constraints relating to the average number of short loops, relaxing the typical requirement for networks to be locally tree like. We review existing results in the literature regarding the Strauss model solved via a series expansion (based on Burda et al. [2004b]), results and techniques of the replica method and the concept of generating a network by projecting a bipartite network to a monopartite network. We extend Burda et al. [2004b] to derive an expression for the Strauss model for sub-critical parameters, and consider the extent to which the Strauss model can capture the required topology and complexity before it reaches the clustered phase. We re-frame the projection of a bipartite network as a protein complex network, where links indicate that a certain protein is part of a certain complex. The projection onto protein space can be naturally interpreted as two proteins being linked if and only if they jointly participate in at least one complex. We propose an approximate graphical expansion to calculating the Shannon entropy. For both the Strauss model and the protein complex model we also formulate the calculation in terms of the replica approach, although this route is not concluded. Table 10.1 lists the results obtained in chapters 3 to 5.

Non-directed Graphs	$\frac{1}{2}\bar{k}[\log(N/\bar{k})+1] - \sum_k p(k)\log\left[\frac{p(k)}{\pi_{\bar{k}}(k)}\right] - \frac{1}{2}\bar{k}\sum_{k,k'} W(k,k')\log\left[\frac{W(k,k')}{W(k)W(k')}\right]$
Directed Graphs	$\bar{k}[\log(N/\bar{k})+1] - \sum_{\vec{k}} p(\vec{k})\log\left[\frac{p(\vec{k})}{\pi_{\bar{k}}(k^{\text{in}})\pi_{\bar{k}}(k^{\text{out}})}\right] - \bar{k}\sum_{\vec{k},\vec{k}'} W(\vec{k},\vec{k}')\log\left[\frac{W(\vec{k},\vec{k}')}{W_1(\vec{k})W_2(\vec{k}')}\right]$
Bipartite Graphs	$\bar{k}\log\left(N/\bar{k}\right) - f\log f - (1-f)\log(1-f) - \sum_k\left[f p_A(k)\log\left(\frac{p_A(k)}{\pi_{\bar{k}}(k)}\right) + (1-f)p_B(k)\log\left(\frac{p_B(k)}{\pi_{\bar{k}}(k)}\right)\right]$
Specified Neighbourhoods	$\frac{1}{2}\bar{k}[\log(N/\bar{k})-1] - \sum_n p(n)\log p(n) + \frac{1}{2}\bar{k}\sum_{k,k'} W(k,k')\log W(k,k') - \sum_n p(n)\sum_k \log\left[\left(\sum_{s\leq k(n)}\delta_{k,\xi^s(n)}\right)!\right]$
Generalised Degrees	$\frac{1}{2}\bar{k}[\log(N/\bar{k})+1] + \frac{1}{2}\bar{k}\sum_{k,k'} W(k,k')\log W(k,k') - \sum_{k,m} p(k,m)\log\left(\frac{p(k,m)}{\pi(k)}\right) + \sum_{k,m} p(k,m)\log\left(\sum_{\xi^1,..,\xi^k}\delta_{m,\sum_{s=1}^k \xi^s}\right)$
Strauss (estimate $\bar{m}\approx O(1)$)	$\frac{1}{2}\bar{k}[\log(N/\bar{k})+1] - \bar{m}\log N + 3\bar{m}\log(\bar{k}-6\bar{m}) - \frac{\bar{k}}{2}\log\left(1-\frac{6\bar{m}}{\bar{k}}\right) - \bar{m}\log\bar{m} - 2\bar{m}$
Strauss (sub-critical region)	$\frac{1}{N}\left[\frac{\bar{k}(N-1)}{2}\left[1+\ln\left(\frac{N}{\bar{k}}\right)\right] - \frac{3\bar{k}^2}{4} + \frac{\langle T\rangle}{6}\left[1+\ln\left(\frac{N^3}{\langle T\rangle}\right)\right] - \frac{\bar{k}^3}{6}\left[1+\ln\left(\frac{N^3}{\bar{k}^3}\right)\right]\right]$
Protein Complex Model (est.)	$\approx \frac{1}{N}\left[\left(1+\frac{q^2}{2}\right)^{\alpha N} \mathrm{e}^{-\alpha q N}\left(\left(\alpha N q + \frac{q^2}{2}\log N\right)\left(1+\frac{q}{3}\right) + \frac{q}{3}\log N\right) - p_0\log p_0\right]$

Table 10.1: A summary of expressions for the leading order entropy per node, as derived in this thesis, for various constrained ensembles. Previously published results in the shaded area for comparison.

In chapter 6 we built on the work of Rao et al. [1996] and Coolen et al. [2009] to define an ergodic and unbiased degree-preserving stochastic process for randomising directed binary non-self-interacting networks, considering the case where the number of moves that can act on each state is variable. The result takes the form of a canonical Markov Chain Monte Carlo (MCMC) algorithm based on simple directed edge swaps and triangle reversals. The acceptance probabilities correct for the entropic bias which is caused by the state dependence of the number of moves that can be executed. Our process is precise for any network size and network topology, and sufficiently versatile to allow random directed graphs with the correct in- and out-degree sequence to be generated with arbitrary desired sampling probabilities. The algorithm can be used to generate more sophisticated null-models which inherit from a real network both the degree sequence and the degree correlations, but are otherwise random and unbiased.

The stated aim of this thesis was to develop a suite of tools to quantify and compare topological properties of cellular signalling networks. We have approached this by extending the range of properties for which we can analyse the entropy of the associated random graph ensemble. This provides a bridge to biology, in that it allows much greater scope for the bioinfomatician to choose and tailor topologies to their needs and intuition. It makes the method more flexible and more relevant. In addition, we have developed a series of illustrations combining the new results and information-theoretic concepts with real biological datasets and problems. We quantified the topology of some gene regulation networks based on degree distribution and degree-degree correlation, and showed that the complexity does not materially change if different confidence thresholds are applied to the data. We quantified the topology of some protein-protein interaction networks under the generalised degree constraint, and showed that this was typically a demanding condition.

We also plot heat-maps of generalised degree distributions of real and synthetic networks, in order to demonstrate that generalised degrees are a much more specific topological signature than simple degrees. Royer et al. [2008] use a degree constrained null model to study the frequency of occurrence of motifs in real networks. In chapter 8 we repeat their analysis, but adding the degree-degree correlation null model which was considered in chapter 6. We find a significant difference in Z-values between the two models, without a clear link between the direction and magnitude of the difference. Motifs non-trivially depend on global topological properties beyond degree distribution - and our results demonstrate that caution is needed before attributing

over-representation of motifs to local or organism-specific factors. For a more extreme and idiosyncratic example of motif density in tailored null models, we define a network of Olympic events, where a link indicates that the two events have at least one competitor in common.

Finally, we take inspiration from Manke et al. [2006] and the lethality-centrality concept of Jeong et al. [2001] in order to develop a new method of ranking nodes based on their contribution to the complexity of the degree-degree correlation topology. This measure is particularly pertinent in the sub-hub region, where degree does not discriminate, since there are typically many nodes with the same degree. This sophistication may be crucial in - for example - choosing drug targets - where knocking out the most highly connected node may have unacceptable unintended consequences. In the sub-hub region (degrees between 5 and 20), we find that our new metric performs substantially better.

The novelty of this work lies in new results derived by established methods and in new methods introduced and developed to increase the applicability of these ideas beyond what was previously possible. This provides a selection of tools and approaches which can be adapted to analyse and inform a range of bioinformatics problems. Bespoke software tools were programmed in C++ to facilitate applications, including features such as network analysis, entropy evaluation, generation of constrained random graphs, Strauss model with higher order terms and the automation of the node ranking procedure described in chapter 9. Highlights of the code architecture and algorithms are outlined in appendix F.

These tools are relevant to the design of network models. The insight of the practitioner can define topological properties which are viewed to be critical to the design and function of the network. A random graph ensemble defined to meet these topological properties provides a pool of null-models - parallel universes where the network could plausibly function in the same way with different topology. Well defined null models provide a valuable check and challenge to numerical work. The Shannon entropy provides a rigorous quantitative measurement of the specificity of a model incorporating these constraints.

The challenge in modelling a real network is to isolate the essential features, but to allow sufficient residual freedom. For any particular network, it would be possible to define a tightening sequence of topological descriptors, which will ultimately define a family of one. It is a stimulating alternative viewpoint to consider a network to be described by its topological constraints, rather than the more traditional connectivity list.

REFERENCES

M. L Acencio and N Lemke. Towards the prediction of essential genes by integration of network topology, cellular localization and biological process information. *BMC Bioinformatics*, 10 (1):290+, 2009.

E Agliari, A Annibale, A Barra, A Coolen, and D Tantari. Immune networks: multi-tasking capabilities at medium load. *Journal of Physics A: Mathematical and Theoretical*, 46(33): 335101, 2013a.

E Agliari, A Annibale, A Barra, A Coolen, and D Tantari. Immune networks: multitasking capabilities near saturation. *Journal of Physics A: Mathematical and Theoretical*, 46(41): 415003, 2013b.

R Albert and A.-L Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

U Alon. *An introduction to systems biology: design principles of biological circuits*. CRC press, 2006.

K Anand and G Bianconi. Entropy measures for networks: Toward an information theory of complex topologies. *Physical Review E*, 80(4):045102, 2009.

K Anand and G Bianconi. Gibbs entropy of network ensembles by cavity methods. *Physical Review E*, 82(1):011116, 2010.

A Annibale and A Coolen. What you see is not what you get: how sampling affects macroscopic features of biological networks. *Interface Focus*, 1(6):836–856, 2011.

A Annibale, A. C Coolen, L. P Fernandes, F Fraternali, and J Kleinjung. Tailored graph ensembles as proxies or null models for real networks I: tools for quantifying structure. *Journal of Physics A: Mathematical and Theoretical*, 42(48):485001, 2009.

M Arifuzzaman, M Maeda, A Itoh, K Nishikata, C Takita, R Saito, T Ara, K Nakahigashi, H.-C Huang, A Hirai, et al. Large-scale identification of protein–protein interaction of Escherichia coli K-12. *Genome research*, 16(5):686–691, 2006.

S Bansal, S Khandelwal, and L Meyers. Exploring biological network structure with clustered random networks. *BMC Bioinformatics*, 10(1):405, 2009.

A.-L Barabási and R Albert. Emergence of scaling in random networks. *Science*, 286(5439): 509–512, 1999.

A.-L Barabasi and Z. N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

R. J Baxter. *Exactly solved models in statistical mechanics*. Courier Dover Publications, 2007.

G Bianconi. Mean field solution of the Ising model on a Barabási–Albert network. *Physics Letters A*, 303(2):166–168, 2002.

G Bianconi. The entropy of randomized network ensembles. *EPL (Europhysics Letters)*, 81(2): 28005, 2008.

G Bianconi. Entropy of network ensembles. *Physical Review E*, 79(3):036114+, 2009.

G Bianconi. Statistical mechanics of multiplex networks: Entropy and overlap. *Physical Review E*, 87(6):062806, 2013.

G Bianconi, A. C Coolen, and C. J. P Vicente. Entropies of complex networks with hierarchically constrained topologies. *Physical Review E*, 78(1):016114, 2008.

G Bianconi, P Pin, and M Marsili. Assessing the relevance of node features for network structure. *Proceedings of the National Academy of Sciences of the United States of America*, 106 (28):11433–11438, 2009.

C Bordenave and P Caputo. Large deviations of empirical neighborhood distribution in sparse random graphs. *arXiv:1308.5725*, 2013.

R Bowley and M Sanchez. *Introductory statistical mechanics*. Clarendon Press Oxford, 1999.

Z Burda. Email Communication, 2012.

Z Burda, J Jurkiewicz, and A Krzywicki. Perturbing general uncorrelated networks. *Physical Review E*, 70(2):026106, 2004a.

Z Burda, J Jurkiewicz, and A Krzywicki. Network transitivity and matrix models. *Physical Review E*, 69:026106, 2004b.

H.-Y Chuang, E Lee, Y.-T Liu, D Lee, and T Ideker. Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 3(1), 2007.

F Chung and L Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2004.

S. R Collins, P Kemmeren, X.-C Zhao, J. F Greenblatt, F Spencer, F. C Holstege, J. S Weissman, and N. J Krogan. Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae. *Molecular & Cellular Proteomics*, 6(3):439–450, 2007a.

S. R Collins, K. M Miller, N. L Maas, A Roguev, J Fillingham, C. S Chu, M Schuldiner, M Gebbia, J Recht, M Shales, et al. Functional dissection of protein complexes involved in yeast chromosome biology using a genetic interaction map. *Nature*, 446(7137):806–810, 2007b.

A. C Coolen, A De Martino, and A Annibale. Constrained Markovian dynamics of random graphs. *Journal of Statistical Physics*, 136:1035–1067, 2009.

S Coulomb, M Bauer, D Bernard, and M.-C Marsolier-Kergoat. Gene essentiality and the topology of protein interaction networks. *Proceedings of the Royal Society B: Biological Sciences*, 272(1573):1721–1725, 2005.

P. G de Gennes. Exponents for the excluded volume problem as derived by the Wilson method. *Physics Letters A*, 38(5):339–340, 1972.

C. I Del Genio, H Kim, Z Toroczkai, and K. E Bassler. Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS ONE*, 5(4):e10012+, 2010.

G del Rio, D Koschützki, and G Coello. How to identify essential genes from molecular networks? *BMC Systems Biology*, 3(1):102, 2009.

L Demetrius and T Manke. Robustness and network evolution–an entropic principle. *Physica A: Statistical and Theoretical Physics*, 346(3-4):682–696, 2005.

S. N Dorogovtsev and J. F Mendes. Evolution of networks. *Advances in Physics*, 51(4):1079–1187, 2002.

S. N Dorogovtsev, A. V Goltsev, and J. F Mendes. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4):1275, 2008.

S Dorogovtsev, J Mendes, and A Samukhin. Multifractal properties of growing networks. *EPL (Europhysics Letters)*, 57(3):334, 2002.

S Edwards and R. C Jones. The eigenvalue spectrum of a large symmetric random matrix. *Journal of Physics A: Mathematical and General*, 9(10):1595, 1976.

H El-Samad, H Kurata, J Doyle, C Gross, and M Khammash. Surviving heat shock: control strategies for robustness and performance. *Proceedings of the National Academy of Sciences of the United States of America*, 102(8):2736–2741, 2005.

P Erdös and A Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5:17–61, 1960.

L Euler. Leonhard Euler and the Königsberg bridges. *Scientific American*, 189:66–70, 1953.

R. M Ewing, P Chu, F Elisma, H Li, P Taylor, S Climie, L McBroom-Cerajewski, M. D Robinson, L O'Connor, M Li, et al. Large-scale mapping of human protein–protein interactions by mass spectrometry. *Molecular systems biology*, 3(1), 2007.

L. P Fernandes, A Annibale, J Kleinjung, A. C Coolen, and F Fraternali. Protein networks reveal detection bias and species consistency when analysed by information-theoretic methods. *PloS ONE*, 5(8):e12083+, 2010.

R Ferrer and R. V Solé. *Optimization in Complex Networks*, pages 114–126. 2003.

R Feynman. Statistical Mechanics, A Set of Lectures,California, Institute of Technology, 1972.

H Fu. *Protein-protein interactions*. Springer, 2004.

A.-C Gavin, P Aloy, P Grandi, R Krause, M Boesche, M Marzioch, C Rau, L. J Jensen, S Bastuck, B Dümpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, 2006.

N. J Gotelli and W Ulrich. Statistical challenges in null model analysis. *Oikos*, 121:171–180, 2012.

J.-D. J Han, N Bertin, T Hao, D. S Goldberg, G. F Berriz, L. V Zhang, D Dupuy, A. J. M Walhout, M. E Cusick, F. P Roth, and M Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, 2004.

C. T Harbison, D. B Gordon, T. I. I Lee, N. J Rinaldi, K. D Macisaac, T. W Danford, N. M Hannett, J.-B. B Tagne, D. B Reynolds, J Yoo, E. G Jennings, J Zeitlinger, D. K Pokholok, M Kellis, P. A Rolfe, K. T Takusagawa, E. S Lander, D. K Gifford, E Fraenkel, and R. A Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104, 2004.

T. R Hughes, M. J Marton, A. R Jones, C. J Roberts, R Stoughton, C. D Armour, H. A Bennett, E Coffey, H Dai, Y. D He, M. J Kidd, A. M King, M. R Meyer, D Slade, P. Y Lum, S. B Stepaniants, D. D Shoemaker, D Gachotte, K Chakraburtty, J Simon, M Bard, and S. H Friend. Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126, 2000.

Y.-C Hwang, C.-C Lin, J.-Y Chang, H Mori, H.-F Juan, and H.-C Huang. Predicting essential genes based on network and sequence analysis. *Molecular BioSystems*, 5(12):1672–1678, 2009.

E Ising. A contribution to the theory of ferromagnetism. *Z. Phys*, 31(1):253–258, 1925.

T Ito, T Chiba, R Ozawa, M Yoshida, M Hattori, and Y Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(8):4569–4574, 2001.

S Itzkovitz, R Milo, N Kashtan, G Ziv, and U Alon. Subgraphs in random networks. *Physical Review E*, 68(2):026127+, 2003.

H Jeong, S. P Mason, A.-L Barabasi, and Z. N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

J Jeong and P Berman. On cycles in the transcription network of Saccharomyces cerevisiae. *BMC Systems Biology*, 2(1):12, 2008.

M Kardar. Statistical Mechanics II Statistical Physics of Fields . MIT OCW, 2013.

H Kim, C. I Del Genio, K. E Bassler, and Z Toroczkai. Constructing and sampling directed graphs with given degree sequences. *New Journal of Physics*, 14(2):023012, 2012.

O. D King. Comment on'Subgraphs in random networks'. *Physical Review E*, 70:058101+, 2004.

H Klein-Hennig and A. K Hartmann. Bias in generation of random graphs. *Physical Review E*, 85(2):026101, 2012.

N. J Krogan, G Cagney, H Yu, G Zhong, X Guo, A Ignatchenko, J Li, S Pu, N Datta, A. P Tikuisis, et al. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

O Kuchaiev, T Milenković, V Memišević, W Hayes, and N Pržulj. Topological network alignment uncovers biological function and phylogeny. *Journal of the Royal Society Interface*, 7 (50):1341–1354, 2010.

D. J LaCount, M Vignali, R Chettier, A Phansalkar, R Bell, J. R Hesselberth, L. W Schoenfeld, I Ota, S Sahasrabudhe, C Kurschner, et al. A protein interaction network of the malaria parasite plasmodium falciparum. *Nature*, 438(7064):103–107, 2005.

B. C Lehne. *Computational analyses of complex diseases at the gene and network levels*. PhD thesis, Department of Medical & Molecular Genetics, 2012.

M Leone, A Vázquez, A Vespignani, and R Zecchina. Ferromagnetic ordering in graphs with arbitrary degree distribution. *The European Physical Journal B-Condensed Matter and Complex Systems*, 28(2):191–197, 2002.

A Lesk. *Introduction to protein science: architecture, function, and genomics*. Oxford University Press, 2010.

A Lesk. *Introduction to genomics*. Oxford University Press, 2012.

T. R Lezon, J. R Banavar, M Cieplak, A Maritan, and N. V Fedoroff. Using the principle of entropy maximization to infer genetic interaction networks from gene expression patterns. *Proceedings of the National Academy of Sciences*, 103(50):19033–19038, 2006.

A Ma and R Mondragón. Evaluation of network robustness using a node tearing algorithm. *Physica A: Statistical Mechanics and its Applications*, 391(24):6674–6681, 2012.

T Manke, L Demetrius, and M Vingron. An entropic characterization of protein interaction networks and cellular robustness. *Journal of The Royal Society Interface*, 3(11):843–850, 2006.

S Maslov and K Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

M Matsumoto and T Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

M Mézard and G Parisi. A replica analysis of the travelling salesman problem. *Journal de Physique*, 47(8):1285–1296, 1986.

M Mézard, G Parisi, and M. A Virasoro. *Spin glass theory and beyond*, volume 9. World scientific Singapore, 1987.

I Miklós and J Podani. Randomization of presence-absence matrices: comments and new algorithms. *Ecology*, 85(1):86–92, 2004.

R Milo, N Kashtan, S Itzkovitz, M. E. J Newman, and U Alon. On the uniform generation of random graphs with prescribed degree sequences . arXiv:cond-mat/0312028, 2004.

R Milo, S Shen-Orr, S Itzkovitz, N Kashtan, D Chklovskii, and U Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

M Molloy and B Reed. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, 6(2-3):161–180, 1995.

M Molloy and B Reed. The size of the giant component of a random graph with a given degree sequence. *Combinatorics, Probability and Computing*, 7(3):295–305, 1998.

M. E Newman. *Networks: an introduction*. OUP Oxford, 2009.

M. E Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

M. E Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

M. E Newman. Properties of highly clustered networks. *Physical Review E*, 68(2):026121, 2003.

T Ohnishi, H Takayasu, and M Takayasu. Network motifs in an inter-firm network. *Journal of Economic Interaction and Coordination*, 2010.

H Orland. Mean-field theory for optimization problems. *Le Journal de Physique Lettres*, 1985.

G. A Pagani and M Aiello. The power grid as a complex network: A survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

M. C Palumbo, A Colosimo, A Giuliani, and L Farina. Functional essentiality from topology features in metabolic networks: a case study in yeast. *FEBS letters*, 579(21):4642–4646, 2005.

G Parisi. *Statistical field theory*, volume 28. Perseus Books New York, 1998.

J. R Parrish, J Yu, G Liu, J. A Hines, J. E Chan, B. A Mangiola, H Zhang, S Pacifico, F Fotouhi, V. J DiRita, et al. A proteome-wide protein interaction map for Campylobacter jejuni. *Genome Biology*, 8(7):R130, 2007.

F Passerini and S Severini. The von Neumann entropy of networks. Technical report, University Library of Munich, Germany, 2008.

R. J Prill, P. A Iglesias, and A Levchenko. Dynamic properties of network motifs contribute to biological network organization. *PLoS Biology*, 3(11):e343, 2005.

J.-C Rain, L Selig, H De Reuse, V Battaglia, C Reverdy, S Simon, G Lenzen, F Petel, J Wojcik, V Schächter, et al. The protein–protein interaction map of Helicobacter pylori. *Nature*, 409 (6817):211–215, 2001.

A. R Rao, R Jana, and S Bandyopadhyay. A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. *Sankhya: The Indian Journal of Statistics, Series A*, 58(2), 1996.

J Rehwinkel, P Natalin, A Stark, J Brennecke, S. M Cohen, and E Izaurralde. Genome-wide analysis of mRNAs regulated by Drosha and Argonaute proteins in Drosophila melanogaster. *Molecular and cellular biology*, 26(8):2965–2975, 2006.

W Ritchie, S Granjeaud, D Puthier, and D Gautheret. Entropy measures quantify global splicing disorders in cancer. *PLoS Comput Biol*, 4(3):e1000011+, 2008.

E. S Roberts, T Schlitt, and A. C Coolen. Tailored graph ensembles as proxies or null models for real networks II: results on directed graphs. *Journal of Physics A Mathematical General*, 44(26):5002, 2011.

L Royer, M Reimann, B Andreopoulos, and M Schroeder. Unraveling protein networks with power graph analysis. *PLoS Computational Biology*, 4(7):e1000108, 2008.

J.-F Rual, K Venkatesan, T Hao, T Hirozane-Kishikawa, A Dricot, N Li, G. F Berriz, F. D Gibbons, M Dreze, N Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173–1178, 2005.

J Rung, T Schlitt, A Brazma, K Freivalds, and J Vilo. Building and analysing genome-wide gene disruption networks. *Bioinformatics (Oxford, England)*, 18 Suppl 2, 2002.

T Schlitt and A Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(Suppl 6):S9+, 2007.

T Schlitt, K Palin, J Rung, S Dietmann, M Lappe, E Ukkonen, and A Brazma. From gene networks to gene function. *Genome Research*, 13(12):2568–2576, 2003.

J. J Seidel. A survey of two-graphs. In A Doe, editor, *In Colloquio Internazionale sulle Teorie Combinatorie Tomo I*, pages 481–511. 1973.

S. S Shen-Orr, R Milo, S Mangan, and U Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature genetics*, 31(1):64–68, 2002.

J. P Sheppard, J.-P Wang, and P. C Wong. Large-scale cortical functional organization and speech perception across the lifespan. *PloS ONE*, 6(1):e16510, 2011.

T Squartini and D Garlaschelli. Exact maximum-likelihood method to detect patterns in real networks. *LEM Working Paper Series*, 2011a.

T Squartini and D Garlaschelli. Analytical maximum-likelihood method to detect patterns in real networks. *New Journal of Physics*, 13(8):083001+, 2011b.

T Squartini, G Fagiolo, and D Garlaschelli. Rewiring World Trade. Part I: A Binary Network Analysis. *LEM Working Paper Series*, 2011.

D Strauss. On a general class of models for interaction. *SIAM Review*, 28(4):513–527, 1986.

L Tabourier, C Roth, and J.-P Cointet. Generating constrained random graphs using multiple edge switches. *Journal of Experimental Algorithmics (JEA)*, 16:1–7, 2011.

K Tarassov, V Messier, C. R Landry, S Radinovic, M. M. S Molina, I Shames, Y Malitskaya, J Vogel, H Bussey, and S. W Michnick. An in vivo map of the yeast protein interactome. *Science*, 320(5882):1465–1470, 2008.

R Taylor. Contrained switchings in graphs. In K McAvaney, editor, *Combinatorial Mathematics VIII*, volume 884 of *Lecture Notes in Mathematics*, pages 314–336. Springer Berlin Heidelberg, 1981.

B Titz, S. V Rajagopala, J Goll, R Häuser, M. T McKevitt, T Palzkill, and P Uetz. The binary protein interactome of Treponema pallidum–the syphilis spirochete. *PLoS ONE*, 3(5):e2292, 2008.

S Uddin, S. T. H Murshed, and L Hossain. Power-law behavior in complex organizational communication networks during crisis. *Physica A: Statistical Mechanics and its Applications*, 390(15):2845–2853, 2011.

P Uetz, L Giot, G Cagney, T. A Mansfield, R. S Judson, J. R Knight, D Lockshon, V Narayan, M Srinivasan, P Pochart, et al. A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae. *Nature*, 403(6770):623–627, 2000.

V Van Kerrebroeck and E Marinari. Ranking vertices or edges of a network by loops: a new approach. *Physical Review Letters*, 101(9):098701+, 2008.

C Von Mering, R Krause, B Snel, M Cornell, S. G Oliver, S Fields, and P Bork. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, 417(6887):399–403, 2002.

B Wang, H Tang, C Guo, and Z Xiu. Entropy optimization of scale-free networks robustness to random failures. *Physica A: Statistical Mechanics and its Applications*, 363(2):591–596, 2006.

S Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

D. J Watts and S. H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393 (6684):440–442, 1998.

J West, G Bianconi, S Severini, and A. E Teschendorff. Differential network entropy reveals cancer system hallmarks. *Scientific reports*, 2, 2012.

J. S Yedidia. Quenched disorder: understanding glasses using a variational principle and the replica method. *Lectures delivered at the Santa Fe Summer School on Complex Systems*, 1992.

S Zhou and R Mondragón. The rich-club phenomenon in the internet topology. *IEEE Communications Letters*, 8(3):180–182, 2004.

E Zotenko, J Mestre, D. P O'Leary, and T. M Przytycka. Why do hubs in the yeast protein interaction network tend to be essential: reexamining the connection between the network topology and essentiality. *PLoS Comput Biol*, 4(8):e1000140+, 2008.

APPENDIX A

SUPPLEMENTARY CALCULATIONS (DIRECTED NETWORKS)

A.1 Order parameter representation of the graph probabilities

In this section we derive a tool that is repeatedly used in chapter 3: a formula in terms of simple observables and order parameters of the log-probability per node of graphs (3.2.6) given the ensemble definition (3.2.1), in leading orders in N. Upon substituting (3.2.1) into this formula, and after some simple manipulations and use of the law of large numbers, one finds

$$\Omega(\boldsymbol{c}|p,Q) = \sum_{\vec{k}} p(\vec{k}|\boldsymbol{c}) \log p(\vec{k}) + \phi_1(\boldsymbol{c}|Q) - \phi_2(\boldsymbol{c}|Q) + \epsilon_N \tag{A.1.1}$$

$$\phi_1(\boldsymbol{c}|Q) = \frac{1}{N} \log w(\boldsymbol{c}|\vec{k}_1,\ldots,\vec{k}_N,Q)\Big|_{\vec{k}_i=\vec{k}_i(\boldsymbol{c})\ \forall i} \tag{A.1.2}$$

$$\phi_2(\boldsymbol{c}|Q) = \frac{1}{N} \log Z(\vec{k}_1,\ldots,\vec{k}_N,Q)\Big|_{\vec{k}_i=\vec{k}_i(\boldsymbol{c})\ \forall i} \tag{A.1.3}$$

with $\epsilon_N \to 0$ as $N \to \infty$, and

$$Z(\vec{k}_1,\ldots,\vec{k}_N,Q) = \sum_{\boldsymbol{c}} w(\boldsymbol{c}|\vec{k}_1,\ldots,\vec{k}_N,Q) \prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \tag{A.1.4}$$

$$w(\boldsymbol{c}|\vec{k}_1,\ldots,\vec{k}_N,Q) = \prod_{i\neq j} \Big[\frac{\overline{k}}{N} Q(\vec{k}_i,\vec{k}_j|\bar{p}) \delta_{c_{ij},1} + \Big(1 - \frac{\overline{k}}{N} Q(\vec{k}_i,\vec{k}_j|\bar{p})\Big) \delta_{c_{ij},0} \Big] \tag{A.1.5}$$

In these expressions $\overline{k} = N^{-1}\sum_i k_i^{\rm in} = N^{-1}\sum_i k_i^{\rm out}$, $\bar{p}(\vec{k}) = N^{-1}\sum_i \delta_{\vec{k},\vec{k}_i}$, and the kernel $Q(.,.)$ is normalized locally according to $\sum_{\vec{k},\vec{k}'}\bar{p}(\vec{k})\bar{p}(\vec{k}')Q(\vec{k},\vec{k}'|\bar{p}) = 1$.

A.1.1 Calculation of ϕ_1

The first contribution (A.1.2) to the entropy is calculated easily:

$$\begin{aligned}\phi_1(\boldsymbol{c}|Q) &= \frac{1}{N}\sum_{i\neq j}\Big\{c_{ij}\log\Big[\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\Big] - \frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\Big\}\Big|_{\vec{k}_i=\vec{k}_i(\boldsymbol{c})\ \forall i} + O(\frac{1}{N}) \\ &= \bar{k}(\boldsymbol{c})\Big\{\log\Big[\frac{\overline{k}(\boldsymbol{c})}{N}\Big] - 1 + \sum_{\vec{k},\vec{k}'}W(\vec{k},\vec{k}'|\boldsymbol{c})\log Q(\vec{k},\vec{k}'|p(.|\boldsymbol{c}))\Big\} + O(\frac{1}{N}) \qquad \text{(A.1.6)}\end{aligned}$$

It involves the in- and out degree distribution $p(\vec{k}|\boldsymbol{c})$, its degree average $\bar{k}(\boldsymbol{c})$, and the joint distribution $W(\vec{k},\vec{k}'|\boldsymbol{c})$ of in- and out degrees of connected nodes. All are calculated for the graph $\boldsymbol{c}$ and defined as

$$p(\vec{k}|\boldsymbol{c}) = \frac{1}{N}\sum_i \delta_{\vec{k},\vec{k}(\boldsymbol{c})} \qquad \text{(A.1.7)}$$

$$W(\vec{k},\vec{k}'|\boldsymbol{c}) = \frac{1}{N\bar{k}(\boldsymbol{c})}\sum_{ij}c_{ij}\delta_{\vec{k},\vec{k}_i(\boldsymbol{c})}\delta_{\vec{k}',\vec{k}_j(\boldsymbol{c})} \qquad \text{(A.1.8)}$$

They are related via the two identities

$$W_1(\vec{k}|\boldsymbol{c}) = \sum_{\vec{k}'}W(\vec{k},\vec{k}'|\boldsymbol{c}) = \frac{k^{\rm in}}{\bar{k}(\boldsymbol{c})}p(\vec{k}|\boldsymbol{c}) \qquad \text{(A.1.9)}$$

$$W_2(\vec{k}|\boldsymbol{c}) = \sum_{\vec{k}'}W(\vec{k}',\vec{k}|\boldsymbol{c}) = \frac{k^{\rm out}}{\bar{k}(\boldsymbol{c})}p(\vec{k}|\boldsymbol{c}) \qquad \text{(A.1.10)}$$

The kernel in (A.1.6) is normalized according to $\sum_{\vec{k},\vec{k}'}p(\vec{k}|\boldsymbol{c})p(\vec{k}'|\boldsymbol{c})Q(\vec{k},\vec{k}'|p(.|\boldsymbol{c})) = 1$.

A.1.2 Calculation of ϕ_2

In order to calculate (A.1.3) we first work out the following quantity, which will then have to be evaluated at $(\vec{k}_1,\ldots,\vec{k}_N) = (\vec{k}_1(\boldsymbol{c}),\ldots,\vec{k}_N(\boldsymbol{c}))$:

$$\begin{aligned}\tilde{\phi}_2(\vec{k}_1,\ldots,\vec{k}_N|Q) &= \frac{1}{N}\log Z(\vec{k}_1,\ldots,\vec{k}_N,Q) \\ &= \frac{1}{N}\log\sum_{\boldsymbol{c}}\prod_{i\neq j}\Big[\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\delta_{c_{ij},1} + \Big(1-\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})\Big)\delta_{c_{ij},0}\Big]\times\prod_i\delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})} \\ &= \frac{1}{N}\log\int_{-\pi}^{\pi}\prod_i\Big[\frac{{\rm d}\omega_i{\rm d}\psi_i}{4\pi^2}{\rm e}^{{\rm i}[\omega_i k_i^{\rm in}+\psi_i k_i^{\rm out}]}\Big]L(\boldsymbol{\omega},\boldsymbol{\psi}|\bar{p},Q) \qquad \text{(A.1.11)}\end{aligned}$$

with

$$\begin{aligned} L(\boldsymbol{\omega},\boldsymbol{\psi}|\bar{p},Q) &= \prod_{i\neq j}\Big[1+\frac{\overline{k}}{N}Q(\vec{k}_i,\vec{k}_j|\bar{p})[\mathrm{e}^{-\mathrm{i}(\omega_i+\psi_j)}-1]\Big] \\ &= \exp\Big[\frac{\overline{k}}{N}\sum_{ij}Q(\vec{k}_i,\vec{k}_j|\bar{p})[\mathrm{e}^{-\mathrm{i}(\omega_i+\psi_j)}-1]+O(N^0)\Big] \end{aligned} \tag{A.1.12}$$

Upon introducing $R(\vec{k}|\boldsymbol{\omega}) = N^{-1}\sum_i \delta_{\vec{k},\vec{k}_i}\mathrm{e}^{-\mathrm{i}\omega_i}$ and $S(\vec{k}|\boldsymbol{\psi}) = N^{-1}\sum_i \delta_{\vec{k},\vec{k}_i}\mathrm{e}^{-\mathrm{i}\psi_i}$, and inserting $\int\prod_{\vec{k}}[\mathrm{d}R(\vec{k})\mathrm{d}S(\vec{k})\ \delta[R(\vec{k})-R(\vec{k}|\boldsymbol{\omega})]\delta[S(\vec{k})-S(\vec{k}|\boldsymbol{\psi})]]$ with δ-functions written in integral form, we can write

$$\begin{aligned} L(\boldsymbol{\omega},\boldsymbol{\psi}|\bar{p},Q) &= \int\prod_{\vec{k}}\Big[\frac{\mathrm{d}R(\vec{k})\mathrm{d}\hat{R}(\vec{k})\mathrm{d}S(\vec{k})\mathrm{d}\hat{S}(\vec{k})}{4\pi^2/N^2}\mathrm{e}^{\mathrm{i}N[\hat{R}(\vec{k})R(\vec{k})+\hat{S}(\vec{k})S(\vec{k})]}\Big]\mathrm{e}^{O(N^0)} \\ &\quad\times\mathrm{e}^{-\mathrm{i}\sum_i[\hat{R}(\vec{k}_i)\mathrm{e}^{-\mathrm{i}\omega_i}+\hat{S}(\vec{k}_i)\mathrm{e}^{-\mathrm{i}\psi_i}]+\overline{k}N\sum_{\vec{k},\vec{k}'}R(\vec{k})Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}')-\overline{k}N} \end{aligned} \tag{A.1.13}$$

Substituting this back into $\tilde{\phi}_2$, and using the law of large numbers, then gives

$$\tilde{\phi}_2(\ldots) = \frac{1}{N}\log\int\prod_{\vec{k}}\Big[\mathrm{d}R(\vec{k})\mathrm{d}\hat{R}(\vec{k})\mathrm{d}S(\vec{k})\mathrm{d}\hat{S}(\vec{k})\Big]\mathrm{e}^{N\Psi[R,\hat{R},S,\hat{S}|\bar{p},Q]+O(\log N)} \tag{A.1.14}$$

where

$$\begin{aligned} \Psi[R,\hat{R},S,\hat{S}|\bar{p},Q] &= \mathrm{i}\sum_{\vec{k}}[\hat{R}(\vec{k})R(\vec{k})+\hat{S}(\vec{k})S(\vec{k})]+\overline{k}\sum_{\vec{k},\vec{k}'}R(\vec{k})Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}')-\overline{k} \\ &\quad+\sum_{\vec{k}}\bar{p}(\vec{k})\Big\{\log\int_{-\pi}^{\pi}\frac{\mathrm{d}\omega}{2\pi}\mathrm{e}^{\mathrm{i}[\omega k^{\mathrm{in}}-\hat{R}(\vec{k})\mathrm{e}^{-\mathrm{i}\omega}]}+\log\int_{-\pi}^{\pi}\frac{\mathrm{d}\psi}{2\pi}\mathrm{e}^{\mathrm{i}[\psi k^{\mathrm{out}}-\hat{S}(\vec{k})\mathrm{e}^{-\mathrm{i}\psi}]}\Big\} \end{aligned} \tag{A.1.15}$$

After doing the remaining integrals over ω and ψ we get

$$\begin{aligned} \Psi[R,\hat{R},S,\hat{S}|\bar{p},Q] &= \mathrm{i}\sum_{\vec{k}}[\hat{R}(\vec{k})R(\vec{k})+\hat{S}(\vec{k})S(\vec{k})]+\overline{k}\sum_{\vec{k},\vec{k}'}R(\vec{k})Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}')-\overline{k} \\ &\quad+\sum_{\vec{k}}\bar{p}(\vec{k})k^{\mathrm{in}}\log[-\mathrm{i}\hat{R}(\vec{k})]+\sum_{\vec{k}}\bar{p}(\vec{k})k^{\mathrm{out}}\log[-\mathrm{i}\hat{S}(\vec{k})] \\ &\quad-\sum_{\vec{k}}\bar{p}(\vec{k})\log(k^{\mathrm{in}}!k^{\mathrm{out}}!) \end{aligned} \tag{A.1.16}$$

For $N\to\infty$ the quantity $\tilde{\phi}_2(\vec{k}_1,\ldots,\vec{k}_N|Q)$ can be evaluated by steepest descent, giving

$$\lim_{N\to\infty}\tilde{\phi}_2(\ldots) = \mathrm{extr}_{R,\hat{R},S,\hat{S}}\Psi[R,\hat{R},S,\hat{S}|\bar{p},Q]$$

Differentiation of Ψ gives the following saddle-point equations:

$$-\mathrm{i}\hat{R}(\vec{k}) = \bar{p}(\vec{k})k^{\mathrm{in}}/R(\vec{k}) = \overline{k}\sum_{\vec{k}'}Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}') \tag{A.1.17}$$

$$-\mathrm{i}\hat{S}(\vec{k}) = \bar{p}(\vec{k})k^{\mathrm{out}}/S(\vec{k}) = \overline{k}\sum_{\vec{k}'}Q(\vec{k}',\vec{k}|\bar{p})R(\vec{k}') \tag{A.1.18}$$

At the saddle-point we deduce that $\sum_{\vec{k},\vec{k}'} R(\vec{k})Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}') = 1$, and that

$$\Psi[R,\hat{R},S,\hat{S}|\bar{p},Q] = -2\bar{k} - \sum_{\vec{k}} \bar{p}(\vec{k}) \log(k^{\rm in}!k^{\rm out}!) + \sum_{\vec{k}} \bar{p}(\vec{k})k^{\rm in} \log\Big[\frac{\bar{p}(\vec{k})k^{\rm in}}{R(\vec{k}|\bar{p},Q)}\Big] + \sum_{\vec{k}} \bar{p}(\vec{k})k^{\rm out} \log\Big[\frac{\bar{p}(\vec{k})k^{\rm out}}{S(\vec{k}|\bar{p},Q)}\Big] \quad \text{(A.1.19)}$$

in which the functions $R(\vec{k}|\bar{p},Q)$ and $S(\vec{k}|\bar{p},Q)$ are the solutions of

$$R(\vec{k}) = \frac{\bar{p}(\vec{k})k^{\rm in}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k},\vec{k}'|\bar{p})S(\vec{k}')}, \qquad S(\vec{k}) = \frac{\bar{p}(\vec{k})k^{\rm out}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k}',\vec{k}|\bar{p})R(\vec{k}')} \quad \text{(A.1.20)}$$

Finally, the quantity (A.1.3) we aim to calculate is defined as the value of $\tilde{\phi}_2(\ldots)$ upon substituting $(\vec{k}_1,\ldots,\vec{k}^N) \to (\vec{k}_1(\boldsymbol{c}),\ldots,\vec{k}_N(\boldsymbol{c}))$. The only occurrences of the sequence $(\vec{k}_1,\ldots,\vec{k}_N)$ in the formula (A.1.19) are in the values of $\bar{p}(\vec{k})$ and $\bar{k}$, so we obtain $\phi_2(\boldsymbol{c}|Q)$ by making in A.1.19 the substitutions $\bar{p}(\vec{k}) \to p(\vec{k}|\boldsymbol{c})$ and $\bar{k} \to \bar{k}(\boldsymbol{c})$. We conclude that

$$\phi_2(\boldsymbol{c}|Q) = -2\tilde{k} - \sum_{\vec{k}} \tilde{p}(\vec{k}) \log(k^{\rm in}!k^{\rm out}!) + \sum_{\vec{k}} \tilde{p}(\vec{k})k^{\rm in} \log\Big[\frac{\tilde{p}(\vec{k})k^{\rm in}}{R(\vec{k}|\tilde{p},Q)}\Big] + \sum_{\vec{k}} \tilde{p}(\vec{k})k^{\rm out} \log\Big[\frac{\tilde{p}(\vec{k})k^{\rm out}}{S(\vec{k}|\tilde{p},Q)}\Big] \quad \text{(A.1.21)}$$

in which $\tilde{p}(\vec{k}) = p(\vec{k}|\boldsymbol{c})$ and $\tilde{k} = \bar{k}(\boldsymbol{c})$, and in which $R(\vec{k}|\tilde{p},Q)$ and $S(\vec{k}|\tilde{p},Q)$ are the solutions of

$$R(\vec{k}) = \frac{\tilde{p}(\vec{k})k^{\rm in}}{\tilde{k}\sum_{\vec{k}'} Q(\vec{k},\vec{k}'|\tilde{p})S(\vec{k}')}, \qquad S(\vec{k}) = \frac{\tilde{p}(\vec{k})k^{\rm out}}{\tilde{k}\sum_{\vec{k}'} Q(\vec{k}',\vec{k}|\tilde{p})R(\vec{k}')} \quad \text{(A.1.22)}$$

A.1.3 Final analytical expression for Ω

The intermediate results given by equations A.1.6 and A.1.21 can now be substituted back into expression (A.1.1), which gives a formula that is seen to depend on $\boldsymbol{c}$ only via $W(\vec{k},\vec{k}'|\boldsymbol{c})$ and $p(\vec{k}|\boldsymbol{c})$

$$\Omega(\boldsymbol{c}|p,Q) = \Big\{ \sum_{\vec{k}} \tilde{p}(\vec{k}) \log p(\vec{k}) + \tilde{k}[1+\log[\tilde{k}/N]] + \sum_{\vec{k}} \tilde{p}(\vec{k}) \log(k^{\rm in}!k^{\rm out}!) - \sum_{\vec{k}} \tilde{p}(\vec{k})k^{\rm in} \log\Big[\frac{\tilde{p}(\vec{k})k^{\rm in}}{R(\vec{k}|\tilde{p},Q)}\Big] - \sum_{\vec{k}} \tilde{p}(\vec{k})k^{\rm out} \log\Big[\frac{\tilde{p}(\vec{k})k^{\rm out}}{S(\vec{k}|\tilde{p},Q)}\Big] + \tilde{k} \sum_{\vec{k},\vec{k}'} \tilde{W}(\vec{k},\vec{k}') \log Q(\vec{k},\vec{k}'|\tilde{p}) \Big\}_{\tilde{W}(\vec{k},\vec{k}')=W(\vec{k},\vec{k}'|\boldsymbol{c}),\ \tilde{p}(\vec{k})=p(\vec{k}|\boldsymbol{c})} + \epsilon_N \quad \text{(A.1.23)}$$

with $\lim_{N\to\infty} \varepsilon_N = 0$, $\tilde{k} = \sum_{\vec{k}} k^{\rm in}\tilde{p}(\vec{k}) = \sum_{\vec{k}} k^{\rm out}\tilde{p}(\vec{k})$, and with the two functions $S(\vec{k}|\tilde{p},Q)$ and $R(\vec{k}|\tilde{p},Q)$ to be extracted from (3.2.8).

A.2 Calculation of the kernel W

For large N the kernel $W(\vec{k},\vec{k}') = (N\bar{k})^{-1}\sum_{ij} c_{ij}\delta_{\vec{k},\vec{k}_i}\delta_{\vec{k}',\vec{k}_j}$ will be self-averaging in the ensemble 3.2.1, i.e. with probability one any graph generated randomly according to 3.2.1 will exhibit the same kernel, modulo finite size effects. Thus we may for $N\to\infty$ calculate $W(\vec{k},\vec{k}')$ as an average over the ensemble 3.2.1

$$
\begin{aligned}
W(\vec{k},\vec{k}') &= \frac{1}{N\bar{k}}\sum_{r\neq s}\sum_{\vec{k}_1\ldots\vec{k}_N}\frac{\delta_{\vec{k},\vec{k}_r}\delta_{\vec{k}',\vec{k}_s}\prod_i p(\vec{k}_i)}{Z(\vec{k}_1\ldots\vec{k}_N,Q)}\sum_{\boldsymbol{c}}\Big[\prod_i \delta_{\vec{k}_i,\vec{k}_i(\boldsymbol{c})}\Big]c_{rs}\\
&\qquad\times\prod_{i\neq j}\Big[\frac{\bar{k}}{N}Q(\vec{k}_i,\vec{k}_j|p)\delta_{c_{ij},1}+\Big(1-\frac{\bar{k}}{N}Q(\vec{k}_i,\vec{k}_j|p)\Big)\delta_{c_{ij},0}\Big]\\
&= \frac{1}{N^2}\sum_{r\neq s}\sum_{\vec{k}_1\ldots\vec{k}_N}\frac{\delta_{\vec{k},\vec{k}_r}\delta_{\vec{k}',\vec{k}_s}\prod_i p(\vec{k}_i)}{Z(\vec{k}_1\ldots\vec{k}_N,Q)}\int_{-\pi}^{\pi}\prod_i\Big[\frac{\mathrm{d}\omega_i\mathrm{d}\psi_i}{4\pi^2}\mathrm{e}^{\mathrm{i}[\omega_i k_i^{\rm in}+\psi_i k_i^{\rm out}]}\Big]\\
&\quad\times Q(\vec{k}_r,\vec{k}_s|p)[\mathrm{e}^{-\mathrm{i}(\omega_r+\psi_s)}[1+O(\frac{1}{N})]\\
&\quad\times\prod_{i\neq j}\Big[1+\frac{\bar{k}}{N}Q(\vec{k}_i,\vec{k}_j|p)[\mathrm{e}^{-\mathrm{i}(\omega_i+\psi_j)}-1]\Big]\\
&= Q(\vec{k},\vec{k}'|p)\sum_{\vec{k}_1\ldots\vec{k}_N}\frac{\prod_i p(\vec{k}_i)}{Z(\vec{k}_1\ldots\vec{k}_N,Q)}\int_{-\pi}^{\pi}\prod_i\Big[\frac{\mathrm{d}\omega_i\mathrm{d}\psi_i}{4\pi^2}\mathrm{e}^{\mathrm{i}[\omega_i k_i^{\rm in}+\psi_i k_i^{\rm out}]}\Big]\\
&\quad\times L(\boldsymbol{\omega},\boldsymbol{\psi}|p,Q)\Big(\frac{1}{N}\sum_r\delta_{\vec{k},\vec{k}_r}\mathrm{e}^{-\mathrm{i}\omega_r}\Big)\Big(\frac{1}{N}\sum_s\delta_{\vec{k}',\vec{k}_s}\mathrm{e}^{-\mathrm{i}\psi_s}\Big)[1+O(\frac{1}{N})]\\
&= Q(\vec{k},\vec{k}'|p)\sum_{\vec{k}_1\ldots\vec{k}_N}\frac{[1+O(\frac{1}{N})]\prod_i p(\vec{k}_i)}{Z(\vec{k}_1\ldots\vec{k}_N,Q)}\int\prod_{\vec{q}}\Big[\frac{\mathrm{d}R(\vec{q})\mathrm{d}\hat{R}(\vec{q})\mathrm{d}S(\vec{q})\mathrm{d}\hat{S}(\vec{q})}{4\pi^2/N^2}\Big]\\
&\quad\times\mathrm{e}^{\mathrm{i}N\sum_{\vec{q}}[\hat{R}(\vec{q})R(\vec{q})+\hat{S}(\vec{q})S(\vec{q})]+\bar{k}N\sum_{\vec{q},\vec{q}'}Q(\vec{q},\vec{q}'|p)R(\vec{q})S(\vec{q}')-\bar{k}N+O(N^0)}\\
&\quad\times R(\vec{k})S(\vec{k}')\prod_i\int_{-\pi}^{\pi}\Big[\frac{\mathrm{d}\omega\mathrm{d}\psi}{4\pi^2}\mathrm{e}^{\mathrm{i}\omega k_i^{\rm in}+\mathrm{i}\psi k_i^{\rm out}-\mathrm{i}\hat{R}(\vec{k}_i)\mathrm{e}^{-\mathrm{i}\omega}-\mathrm{i}\hat{S}(\vec{k}_i)\mathrm{e}^{-\mathrm{i}\psi}}\Big]
\end{aligned}
\tag{A.2.1}
$$

We now write $Z(\vec{k}_1\ldots\vec{k}_N,Q)$ also as an integral over order parameters, as in our earlier derivation of (A.1.19), but noting that now the relevant degree distribution is that of our ensemble (3.2.1), i.e. $p(\vec{k})$ instead of $\bar{p}(\vec{k})$. This gives

$$
\begin{aligned}
W(\vec{k},\vec{k}') &= [1+O(\frac{1}{N})]Q(\vec{k},\vec{k}')\sum_{\vec{k}_1\ldots\vec{k}_N}\prod_i p(\vec{k}_i)\\
&\quad\times\frac{\int\prod_{\vec{q}}\mathrm{d}R(\vec{q})\mathrm{d}\hat{R}(\vec{q})\mathrm{d}S(\vec{q})\mathrm{d}\hat{S}(\vec{q})\;\mathrm{e}^{N\Psi[R,\hat{R},S,\hat{S}|p,Q]+O(\log N)}R(\vec{k})S(\vec{k}')}{\int\prod_{\vec{q}}\mathrm{d}R(\vec{q})\mathrm{d}\hat{R}(\vec{q})\mathrm{d}S(\vec{q})\mathrm{d}\hat{S}(\vec{q})\;\mathrm{e}^{N\Psi[R,\hat{R},S,\hat{S}|p,Q]+O(\log N)}}
\end{aligned}
\tag{A.2.2}
$$

where the non-extensive terms in the exponentials of numerator and denominator are fully identical, and with Ψ as defined in (A.1.15), modulo the replacement $\bar{p}\to p$. The summation over

degree sequences has now become obsolete, and for $N \to \infty$ we obtain

$$\lim_{N\to\infty} W(\vec{k},\vec{k}') \quad = \quad R(\vec{k}|p,Q)Q(\vec{k},\vec{k}'|p)S(\vec{k}'|p,Q) \tag{A.2.3}$$

in which $R(\vec{k}|p,Q)$ and $S(\vec{k}|p,Q)$ are to be solved from

$$R(\vec{k}) = \frac{p(\vec{k})k^{\text{in}}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k},\vec{k}'|p)S(\vec{k}')}, \qquad S(\vec{k}) = \frac{p(\vec{k})k^{\text{out}}}{\bar{k}\sum_{\vec{k}'} Q(\vec{k}',\vec{k}|p)R(\vec{k}')} \tag{A.2.4}$$

with the average degree of our ensemble, $\bar{k} = \sum_{\vec{k}} k^{\text{in}} p(\vec{k}) = \sum_{\vec{k}} k^{\text{out}} p(\vec{k})$.

APPENDIX B

GENERALISED DEGREE CORRELATION KERNEL FOR ENSEMBLES WITH PRESCRIBED GENERALISED DEGREES

The generalised quantity $W(k,m;k',m')$ in the ensemble with prescribed generalised degree distributions $p(k,m)$ can be calculated along the same lines as the calculation of $W(k,k')$ in the main text. It is defined as

$$W(k,m;k',m'|\boldsymbol{c}) \quad = \quad \frac{1}{N\bar{k}}\sum_{ij} c_{ij}\delta_{k,\sum_\ell c_{i\ell}}\delta_{k',\sum_\ell c_{j\ell}}\delta_{m,\sum_\ell c_{i\ell}k_\ell}\delta_{m',\sum_\ell c_{j\ell}k_\ell} \tag{B.0.1}$$

and its ensemble average takes the form

$$\begin{aligned} W(k,m;k',m') &= \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\, \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}}\left(\frac{1}{N^2}\sum_{rs}\delta_{k,k_r}\delta_{k',k_s}\delta_{m,m_r}\delta_{m',m_s}\mathrm{e}^{-\mathrm{i}(\theta_r+\theta_s+\phi_r k'+\phi_s k)}\right)}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\, \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}}} + O(\frac{1}{N}) \\ &= \frac{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\, \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}}\left(\frac{1}{N}\sum_r\delta_{k,k_r}\delta_{m,m_r}\mathrm{e}^{-\mathrm{i}(\theta_r+\phi_r k')}\right)\left(\frac{1}{N}\sum_s\delta_{k',k_s}\delta_{m',m_s}\mathrm{e}^{-\mathrm{i}(\theta_s+\phi_s k)}\right)}{\int_\pi^\pi \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{\phi}\, \mathrm{e}^{\mathrm{i}(\boldsymbol{\theta}\cdot\boldsymbol{k}+\boldsymbol{\phi}\cdot\boldsymbol{m})+\frac{\bar{k}}{2N}\sum_{ij}\mathrm{e}^{-\mathrm{i}(\theta_i+\theta_j+\phi_i k_j+\phi_j k_i)}}} + O(\frac{1}{N}) \end{aligned}$$

Now we will want to introduce a generalised order parameter, namely

$$P(\theta,\phi,k,m) = \frac{1}{N}\sum_r \delta_{k,k_r}\delta_{m,m_r}\delta(\theta-\theta_r)\delta(\phi-\phi_r) \tag{B.0.2}$$

The previous order parameter used in the calculation of $W(k,k')$ is a marginal of this, via $P(\theta,\phi,k) = \sum_m P(\theta,\phi,k,m)$. This definition will give us

$$W(k,m;k',m') = \Big(\int_{-\pi}^{\pi} \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k,m)\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k'}\Big)\Big(\int_{-\pi}^{\pi} \mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k',m')\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k}\Big) \tag{B.0.3}$$

$$W(k,k') = \sum_{mm'} W(k,m;k',m') \tag{B.0.4}$$

in which the new order parameter and its conjugate are to be solved by extremisation of the generalised surface

$$\begin{aligned}\Psi[P,\hat{P}] = {} & \mathrm{i}\sum_{km}\int_{-\pi}^{\pi}\mathrm{d}\theta\mathrm{d}\phi\ \hat{P}(\theta,\phi,k,m)P\theta,\phi,k,m) \\ & + \sum_{km} P(k,m)\log\int_{-\pi}^{\pi}\mathrm{d}\theta\mathrm{d}\phi\ \mathrm{e}^{\mathrm{i}(\theta k+\phi m-\hat{P}(\theta,\phi,k,m))} \\ & + \frac{1}{2}\bar{k}\int \mathrm{d}\theta\mathrm{d}\phi\mathrm{d}\theta'\mathrm{d}\phi' \sum_{kk'mm'} P(\theta,\phi,k,m)P(\theta',\phi',k',m')\mathrm{e}^{-\mathrm{i}(\theta+\theta'+\phi k'+\phi' k)}\end{aligned} \tag{B.0.5}$$

Variation of Ψ gives the following saddle-point equations

$$\hat{P}(\theta,\phi,k,m) = \mathrm{i}\bar{k}\mathrm{e}^{-\mathrm{i}\theta}\int \mathrm{d}\theta'\mathrm{d}\phi' \sum_{k'm'} P(\theta',\phi',k',m')\mathrm{e}^{-\mathrm{i}(\theta'+\phi k'+\phi' k)} \tag{B.0.6}$$

$$P(\theta,\phi,k,m) = P(k,m)\frac{\mathrm{e}^{\mathrm{i}(\theta k+\phi m-\hat{P}(\theta,\phi,k,m))}}{\int_{-\pi}^{\pi}\mathrm{d}\theta'\mathrm{d}\phi'\ \mathrm{e}^{\mathrm{i}(\theta' k+\phi' m-\hat{P}(\theta',\phi',k,m))}} \tag{B.0.7}$$

Clearly $\hat{P}(\theta,\phi,k,m) = \hat{P}(\theta,\phi,k)$ (i.e. it is independent of m). We may therefore substitute $\hat{P}(\theta,\phi,k) = \mathrm{i}\bar{k}\mathrm{e}^{-\mathrm{i}\theta}\hat{P}(\phi,k)$ and find

$$\hat{P}(\phi,k) = \int \mathrm{d}\theta'\mathrm{d}\phi' \sum_{k'm'} P(\theta',\phi',k',m')\mathrm{e}^{-\mathrm{i}(\theta'+\phi k'+\phi' k)} \tag{B.0.8}$$

$$P(\theta,\phi,k,m) = P(k,m)\frac{\mathrm{e}^{\mathrm{i}(\theta k+\phi m)+\bar{k}\mathrm{e}^{-\mathrm{i}\theta}\hat{P}(\phi,k))}}{\int_{-\pi}^{\pi}\mathrm{d}\theta'\mathrm{d}\phi'\ \mathrm{e}^{\mathrm{i}(\theta' k+\phi' m)+\bar{k}\mathrm{e}^{-\mathrm{i}\theta'}\hat{P}(\phi',k))}} \tag{B.0.9}$$

We observe as before in Coolen et al. [2009] that

$$\begin{aligned}
\int_{-\pi}^{\pi}\mathrm{d}\theta\ P(\theta,\phi,k)\mathrm{e}^{-\mathrm{i}\theta} &= \sum_m P(k,m)\frac{\int_{-\pi}^{\pi}\mathrm{d}\theta\ \mathrm{e}^{\mathrm{i}(\theta(k-1)+\phi m)+\bar{k}\mathrm{e}^{-\mathrm{i}\theta}\hat{P}(\phi,k))}}{\int_{-\pi}^{\pi}\mathrm{d}\theta\mathrm{d}\phi'\ \mathrm{e}^{\mathrm{i}(\theta k+\phi' m)+\bar{k}\mathrm{e}^{-\mathrm{i}\theta}\hat{P}(\phi',k))}}\\
&= \sum_m P(k,m)\frac{\sum_{\ell\geq 0}\frac{\bar{k}^\ell\hat{P}^\ell(\phi,k)}{\ell!}\int_{-\pi}^{\pi}\mathrm{d}\theta\ \mathrm{e}^{\mathrm{i}(\theta(k-1-\ell)+\phi m)}}{\sum_{\ell\geq 0}\frac{\bar{k}^\ell\hat{P}^\ell(\phi',k)}{\ell!}\int_{-\pi}^{\pi}\mathrm{d}\theta\mathrm{d}\phi'\ \mathrm{e}^{\mathrm{i}(\theta k+\phi' m-\ell\theta)}}\\
&= \sum_m P(k,m)\frac{\frac{\bar{k}^{k-1}\hat{P}^{k-1}(\phi,k)}{(k-1)!}\mathrm{e}^{\mathrm{i}\phi m}}{\frac{\bar{k}^k\hat{P}^k(\phi',k)}{k!}\int_{-\pi}^{\pi}\mathrm{d}\phi'\ \mathrm{e}^{\mathrm{i}\phi' m}}\\
&= \sum_m \frac{k}{\bar{k}}P(k,m)\frac{\hat{P}^{k-1}(\phi,k)\mathrm{e}^{\mathrm{i}\phi m}}{\int_{-\pi}^{\pi}\mathrm{d}\phi'\ \hat{P}^k(\phi',k)\mathrm{e}^{\mathrm{i}\phi' m}} \qquad \text{(B.0.10)}
\end{aligned}$$

Hence

$$\hat{P}(\phi,k) = \sum_{k'm'}\frac{k'}{\bar{k}}P(k',m')\mathrm{e}^{-\mathrm{i}\phi k'}\frac{\int_{-\pi}^{\pi}\mathrm{d}\phi'\ \hat{P}^{k'-1}(\phi',k')\mathrm{e}^{\mathrm{i}\phi'(m'-k)}}{\int_{-\pi}^{\pi}\mathrm{d}\phi'\ \hat{P}^{k'}(\phi',k')\mathrm{e}^{\mathrm{i}\phi' m'}} \qquad \text{(B.0.11)}$$

After writing $\hat{P}(\phi,k)=\sum_{k'}\gamma(k,k')\mathrm{e}^{-\mathrm{i}\phi k'}$ we recover our familiar equation

$$\gamma(k,k')\gamma(k',k) = \frac{k'}{\bar{k}}\sum_m P(k',m)\frac{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m,\sum_{n\leq k'}k_n}\delta_{kk_n}}{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m,\sum_{n\leq k'}k_n}} \qquad \text{(B.0.12)}$$

But now we can also work out the generalised kernel

$$\begin{aligned}
W(k,m;k',m') =& \Big(\int_{-\pi}^{\pi}\mathrm{d}\theta d\phi\ P(\theta,\phi,k,m)\mathrm{e}^{-\mathrm{i}\theta-\mathrm{i}\phi k'}\Big)\Big(\int_{-\pi}^{\pi}\mathrm{d}\theta\mathrm{d}\phi\ P(\theta,\phi,k',m')\mathrm{e}^{-\mathrm{i}\theta-i\phi k}\Big)\\
=& \frac{kk'}{\bar{k}^2}P(k,m)P(k',m')\Big(\frac{\int_{-\pi}^{\pi}d\phi\ \hat{P}^{k-1}(\phi,k)\mathrm{e}^{\mathrm{i}\phi(m-k')}}{\int_{-\pi}^{\pi}\mathrm{d}\phi\ \hat{P}^k(\phi,k)\mathrm{e}^{\mathrm{i}\phi m}}\Big)\Big(\frac{\int_{-\pi}^{\pi}\mathrm{d}\phi\ \hat{P}^{k'-1}(\phi,k')\mathrm{e}^{\mathrm{i}\phi(m'-k)}}{\int_{-\pi}^{\pi}\mathrm{d}\phi\ \hat{P}^{k'}(\phi,k')\mathrm{e}^{\mathrm{i}\phi m'}}\Big)\\
=& \frac{kk'}{\bar{k}^2}\frac{P(k,m)P(k',m')}{\gamma(k,k')\gamma(k',k)}\Big(\frac{\sum_{k_1\ldots k_k}\Big[\prod_{n=1}^{k}\gamma(k,k_n)\Big]\delta_{m,\sum_{n\leq k}k_n}\delta_{k_k,k'}}{\sum_{k_1\ldots k_k}\Big[\prod_{n=1}^{k}\gamma(k,k_n)\Big]\delta_{m,\sum_{n\leq k}k_n}}\Big)\\
&\times\Big(\frac{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m',\sum_{n\leq k'}k_n}\delta_{k_{k'},k}}{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m',\sum_{n\leq k'}k_n}}\Big) \qquad \text{(B.0.13)}
\end{aligned}$$

We know that $W(k,k')=\gamma(k,k')\gamma(k',k)$, and that $P(k,m)k/\bar{k}=W(k,m)$, so this can be simplified to

$$\begin{aligned}
W(k,m;k',m') &= \frac{W(k,m)W(k',m')}{W(k,k')}\Big(\frac{\sum_{k_1\ldots k_k}\Big[\prod_{n=1}^{k}\gamma(k,k_n)\Big]\delta_{m,\sum_{n\leq k}k_n}\delta_{k_k,k'}}{\sum_{k_1\ldots k_k}\Big[\prod_{n=1}^{k}\gamma(k,k_n)\Big]\delta_{m,\sum_{n\leq k}k_n}}\Big)\\
&\times\Big(\frac{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m',\sum_{n\leq k'}k_n}\delta_{k_{k'},k}}{\sum_{k_1\ldots k_{k'}}\Big[\prod_{n=1}^{k'}\gamma(k',k_n)\Big]\delta_{m',\sum_{n\leq k'}k_n}}\Big) \qquad \text{(B.0.14)}
\end{aligned}$$

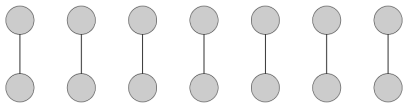
APPENDIX C

DIRECT ENUMERATION OF SOME SIMPLE GENERALISED DEGREE ENSEMBLES

The entropy expression must satisfy certain properties. We can use these to check if our claimed answer (equation 4.5.10) is plausible. Where there is a simple enough example to be enumerated directly, we expect to be able to reconcile the analytical expression for Z with counting $\sum_c \delta_{\vec{k},\vec{k(c)}}$ Where the self consistency relation for γ can be solved directly, we expect to be able to reconcile the implicit and explicit forms of the Γ parameter.

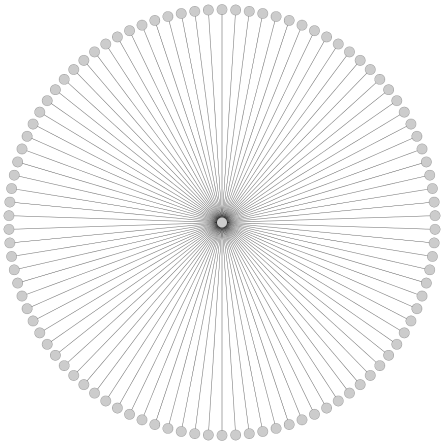
C.1 Ladder configuration

For this case (illustrated in figure C.1), we find that $\gamma(1,1) = 1, W(1,1) = 1$. $\Gamma = 0$ calculated either route. To calculate S from first principles for this network we need to calculate $p(\boldsymbol{c}) = \frac{\prod_i p(k_i)}{Z}$ where we have defined our ensemble as drawing degrees from $p(k) = \delta_{k,1}$. As always, the core of the problem is in calculating $Z(\boldsymbol{k}) = \sum_c \prod_i \delta_{1,k_i(c)}$ which is equivalent to counting the permutations of the network drawn above. There are $N!$ orderings of the nodes, but this



k	1
m	1
$p(k,m)$	1

Figure C.1: A synthetic example with only one degree value - the ladder. In section C.1 the partition function is evaluated directly and compared to the result using equation 4.5.10.



k	1	99
m	99	99
$p(k,m)$	99/100	1/100

Figure C.2: A synthetic example with two degree values - the wheel. In section C.2 the partition function is evaluated directly and compared to the result using equation 4.5.10.

has to be divided by 2 since the bonds are symmetric and by $\frac{N}{2}$ since the order of the bonds is immaterial.

$$\sum_{\boldsymbol{c}} p(\boldsymbol{c}) \log(p(\boldsymbol{c})) = -\log \frac{N!}{2^{\frac{N}{2}}!} = -\log N! + \log 2 + \log \frac{N}{2}! \tag{C.1.1}$$

which by Stirling's approximation goes to $= \frac{-N}{2}(\log N - 1)$ which corresponds to our analytical result in leading order in N.

C.2 Wheel configuration

Consider an ensemble with generalised degrees consistent with figure C.2. Based on the self consistency equations we find that $\gamma(1,s)\gamma(s,1) = \frac{1}{2}$ and so it follows

$$\Gamma = \frac{99}{100}\log\gamma(1,s) + \frac{99}{100}\log\gamma(s,1) = \frac{99}{100}\log\gamma(1,s)\gamma(s,1) = -\frac{\bar{k}}{2}\log 2 \tag{C.2.1}$$

We can immediately deduce the key parameters of this problem worked through as a starting point for an ensemble with given generalised degrees.

$$\bar{k} = 2 \times 99/100 \quad p(k,m) = \frac{\delta_{(k,m),(99,99)}}{100} + \frac{99\delta_{(k,m),(1,99)}}{100} \tag{C.2.2}$$

Let us work out $Z(\boldsymbol{k})$ for an arbitrary degree sequence drawn from our distribution. For the generalised degree case the only valid wiring is as illustrated: groups of 99 degree 1 nodes around a degree 99 node. To enumerate the ensemble, view the peripheral nodes as consecutively numbered; observe $\frac{99N}{100}!$ orderings of such a numbering - of which there are 99! equivalent permutations within each grouping. Hence, claim $Z(\boldsymbol{k}) = \frac{(99N/100)!}{99!^{N/100}}$. So, directly calculated

$$\frac{1}{N}\log Z|_{enumerated} = \frac{99}{100}\left(\log(N) + \log(99/100) - \frac{1}{100}\log 99!\right) - \frac{99}{100} \tag{C.2.3}$$

whereas picking up from the appropriate point in the calculation finds this quantity to be

$$\frac{1}{N}\log Z|_{analytical} = \frac{\bar{k}}{2}(1 + \log(\frac{N}{\bar{k}})) + \sum p(k,m)\log\pi(k) + \Gamma \tag{C.2.4}$$

The $\pi(k)$ factors evaluate to

$$\begin{aligned}\log(\pi(1)) &= \log(\frac{2\times 99}{100}) - \bar{k} \\ \log(\pi(99)) &= \log\left[\frac{2\times 99}{100}\right]^{99} - \bar{k} - \log 99!\end{aligned} \tag{C.2.5}$$

so, inserting equation C.2.1 into C.2.3 shows that

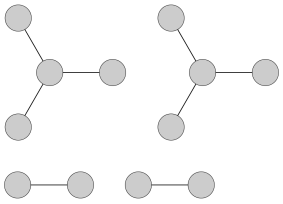
$$\frac{1}{N}\log Z|_{analytical} = -\frac{\bar{k}}{2} + \frac{\bar{k}}{2}\log N + \frac{\bar{k}}{2}\log\bar{k} - \frac{1}{100}\log 99! + \Gamma \tag{C.2.6}$$

which reconciles with the solution from first principles, as required.

C.3 A validation example with a more complicated degree-degree correlation

Consider a network based on figure C.3 as a repeating motif. In fact, this is the only possible wiring of such a network that fulfils the generalised degree distribution required, up to permutation of nodes. As previously, it is convenient to work with

$$\gamma^{\star}(\vec{k'},k) = \frac{\sum_{\{\xi_1\ldots\xi_{k'-1}\}}\prod_{s=1}^{k'-1}\gamma(k',\xi_s)\delta_{m'-k,\sum\xi_s}}{\sum_{\{\xi_1\ldots\xi_{k'}\}}\prod_{s=1}^{k'}\gamma(k',\xi_s)\delta_{m',\sum\xi_s}} \tag{C.3.1}$$



k	1	1	3
m	1	3	3
$p(k,m)$	1/3	1/2	1/6

Figure C.3: A synthetic example with non-constant degree-degree correlation. In section C.3 the partition function is evaluated directly and compared to the result using equation 4.5.10.

satisfying $\sum_{m'} \frac{k'}{\bar{k}} p(\vec{k'})\gamma^\star(\vec{k'},k) = \gamma(k,k')$

But the example has been constructed so that (up to permutation) only one sequence is graphical: $\{(1,1),(1,1),(1,3),(1,3),(1,3),(3,3)....\}$ repeated $N/6$ times. We can do the calculations for the small network only without loss of generality.

The self consistency relations are trivial in this example

$$\begin{aligned} \gamma^2(1,1) &= \frac{p(1,1)}{\bar{k}} = \frac{1}{4} \\ \gamma(1,3)\gamma(3,1) &= \frac{3p(3,3)}{\bar{k}} = \frac{3}{8} \end{aligned} \tag{C.3.2}$$

from which it follows that Γ evaluated from equation 4.5.1 is

$$\begin{aligned} \Gamma &= \frac{\log\gamma(1,1)}{3} + \frac{\log\gamma(1,3)}{2} + \frac{\log\gamma^3(3,1)}{6} \\ &= \frac{1}{6}\log\left[\frac{1}{4}\left(\frac{3}{8}\right)^3\right] \end{aligned} \tag{C.3.3}$$

Which evaluates to the same as the expression for Γ proposed in equation 4.5.10

$$\begin{aligned} \Gamma &= \frac{\bar{k}}{2}\sum_{k,k'} W(k,k')\log W(k,k') + \sum_{k,m} p(k,m)\log\binom{m-1}{k-1} \\ &= \frac{8}{6}\frac{1}{2}\left[\frac{1}{4}\log\frac{1}{4} + \frac{3}{8}\log\frac{3}{8} + \frac{3}{8}\log\frac{3}{8}\right] + 0 \\ &= \frac{1}{6}\log\left[\frac{1}{4}\left(\frac{3}{8}\right)^3\right] \end{aligned} \tag{C.3.4}$$

as predicted. A combinatorial argument can be used to calculate $Z(\boldsymbol{k})$ in this case. Consider the network to be a combination of two sub networks. The sub-network of nodes degree $(1,1)$ has $N/3$ nodes. Repeat the earlier strategy of counting the configurations by counting the number of labellings of the diagram $(\frac{N}{3}!)$, and then dividing by the symmetries: $2!^{N/6}$ since the pairs

are symmetric and $(N/6)!$ since the order does not matter. In the other subnetwork, there are $(N/2)!$ orderings of the peripheral nodes, divided by $(3!)^{N/6}$ symmetry factor (but the 'order' does matter, since this fixes the central node). Hence claim

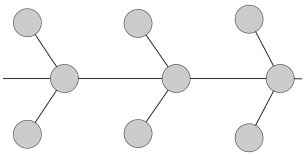
$$Z(\boldsymbol{k}) = \frac{\frac{N}{3}!\frac{N}{2}!}{\frac{N}{6}!2!^{N/6}3!^{N/6}} \tag{C.3.5}$$

If $N = 6$, then $Z(\boldsymbol{k}) = \frac{2!3!}{1!2!3!} = 1$ as expected. If $N = 12$, then $Z(\boldsymbol{k}) = \frac{4!6!}{2!2!2!3!3!} = \frac{6!}{2!3!} = 60$ which can be corroborated directly. Working from first principles, it follows that

$$\begin{aligned} \frac{1}{N}\log Z(\boldsymbol{k}) &= \frac{1}{3}\log\left(\frac{N}{3}\right) + \frac{1}{2}\log\left(\frac{N}{2}\right) - \frac{1}{6}\log\left(\frac{N}{6}\right) - \frac{1}{6}\log(12) - \frac{4}{6} \\ &= \frac{4}{6}\log N - \frac{1}{3}\left[\log 3 + 2\log 2\right] - \frac{2}{3} \end{aligned} \tag{C.3.6}$$

This matches the analytical result, evaluated with the help of equation C.3.4.

C.4 A connected network example



k	1	4
m	4	8
$p(k,m)$	2/3	1/3

Figure C.4: A synthetic connected network example. In section C.4 the partition function is evaluated directly and compared to the result using equation 4.5.10.

Consider the ensemble of networks defined by the general degree sequence set out in figure C.4. By combinatorics, argue that there are $\frac{N}{3}!$ orderings of the centre nodes, and $\frac{2N}{3}!$ orderings of the remaining nodes, divided by $2^{\frac{N}{3}}$ for symmetry. Hence it follows that

$$\begin{aligned} \frac{1}{N}\log Z(\boldsymbol{k},\boldsymbol{m}) &= \frac{1}{N}\log\left(\frac{\frac{N}{3}!\frac{2N}{3}!}{2^{\frac{N}{3}}}\right) \\ &= \frac{1}{3}\log\frac{N}{3} + \frac{2}{3}\log\frac{2N}{3} - \frac{1}{3}\log 2 - 1 \\ &= \log N - \log 3 + \frac{1}{3}\log 2 - 1 \end{aligned} \tag{C.4.1}$$

Working using the analytical formula we can immediately deduce that

$$\frac{1}{N}\log Z(\boldsymbol{k},\boldsymbol{m}) = \log N + \log 2 - 1 - \frac{2}{3}\log 2 - \log 3 \tag{C.4.2}$$

$$= \log N + \frac{1}{3}\log 2 - \log 3 - 1 \tag{C.4.3}$$

APPENDIX D

SUPPLEMENTARY CALCULATIONS (RANDOMISING DIRECTED GRAPHS)

D.1 Efficient calculation of changes in mobility terms following one move

Calculating the mobility $n(\boldsymbol{c})$ terms is computationally heavy. Given that our moves are simple and standard, we follow the alternative route in Coolen et al. [2009] and derive formulae for calculating the *change* in mobility due to one move, so that we can avoid repeated matrix multiplications at each time step.

D.1.1 Change in $n_{\square}(\boldsymbol{c})$ following one square-type move

Without loss of generality, define our square move to be the transformation between matrix $\boldsymbol{c}$ and $\boldsymbol{x}$, involving four nodes (a, b, c, d), such that for all (i, j): $x_{ij} = c_{ij} + \Delta_{ij}$, with

$$\Delta_{ij} \quad = \quad \delta_{ia}\delta_{jd} + \delta_{ic}\delta_{jb} - \delta_{ia}\delta_{jb} - \delta_{ic}\delta_{jd} \tag{D.1.1}$$

We now determine the overall change induced in $n_{\square}(\boldsymbol{c})$ by finding the impact of an edge swap on each term in equation 6.1.9. on the right hand side of the expression above.

- Term 1:

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{x}\boldsymbol{x}^{\dagger}\boldsymbol{x}\boldsymbol{x}^{\dagger}) - \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) &= \sum_{ijkm}\left[c_{ij}c_{kj}c_{km}c_{im} - (c_{ij}+\Delta_{ij})(c_{kj}+\Delta_{kj})(c_{km}+\Delta_{km})(c_{im}+\Delta_{im})\right] \\ &= \Delta_{ij}c_{kj}c_{km}c_{im} + ... + \Delta_{ij}\Delta_{kj}c_{km}c_{im} + ... + \Delta_{ij}\Delta_{kj}\Delta_{km}c_{im} + ... + \Delta_{ij}\Delta_{kj}\Delta_{km}\Delta_{im}\end{aligned}$$

  where ... refers in each case to three similar terms (with their appropriate indices). Let us inspect what happens when two Δ terms are multiplied together. We might have the first suffix repeated, the second suffix repeated, or no repeated sufficies:

$$\begin{aligned}\Delta_{ij}\Delta_{im} &= 2[\delta_{jd}(\delta_{md} - \delta_{mb}) + \delta_{jb}(\delta_{mb} - \delta_{md})] \\ \Delta_{ij}\Delta_{kj} &= 2[\delta_{ia}(\delta_{ka} - \delta_{kc}) + \delta_{ic}(\delta_{kc} - \delta_{ka})]\end{aligned} \tag{D.1.2}$$

  One immediately observes that

$$\sum_{ijkm}\Delta_{ij}\Delta_{kj}\Delta_{km}\Delta_{im} = 4\sum_{ik}[\delta_{ia}\delta_{ia}(\delta_{ka}\delta_{ka} + \delta_{kc}\delta_{kc}) + \delta_{ic}\delta_{ic}(\delta_{kc}\delta_{kc} + \delta_{ka}\delta_{ka})] = 16$$

  To handle two Δ terms with different sufficies we use

$$\Delta_{ij}c_{kj} = c_{kb}\,(\delta_{ic} - \delta_{ia}) + c_{kd}\,(\delta_{ia} - \delta_{ic}) \tag{D.1.3}$$

  which leads us to $\sum_{ijkm}\Delta_{ij}c_{kj}\Delta_{km}c_{im} = 4$. Returning to the result D.1.2 it follows that

$$\Delta_{ij}\Delta_{kj}c_{im}c_{km} = 2\,(\delta_{ia}(\delta_{ka} - \delta_{kc}) + \delta_{ic}(\delta_{kc} - \delta_{ka}))\,c_{im}c_{km} = 2(k_a^{\mathrm{in}} + k_c^{\mathrm{in}}) - 4c_{am}c_{cm}$$

  and the symmetric term gives

$$\Delta_{ij}\Delta_{im}c_{kj}c_{km} \quad = 2(k_d^{\mathrm{out}} + k_b^{\mathrm{out}}) - 4c_{id}c_{ib} \tag{D.1.4}$$

  For the third order terms we combine equations D.1.2 and D.1.3

$$\begin{aligned}\sum_{ijkm}\Delta_{ij}\Delta_{im}\Delta_{kj}c_{km} = \;& 2\sum_{ik}\Big[\,[\delta_{ia}(\delta_{ka}-\delta_{kc})+\delta_{ic}(\delta_{kc}-\delta_{ka})]\,[\delta_{ia}c_{kd}+\delta_{ic}c_{kb}-\delta_{ia}c_{kb}-\delta_{ic}c_{kd}]\,\Big] \\ = \;& 2(c_{ad}-c_{cd}-c_{ab}+c_{cb}-c_{ab}+c_{cb}+c_{ad}-c_{cd}) = -8\end{aligned}$$

By permutation of sufficies all such terms evaluate to -8. Finally we turn to the four terms where only one Δ appears, corresponding to permutations of $\Delta_{ij}c_{kj}c_{km}c_{im} = c_{kd}c_{km}c_{am} + c_{kb}c_{km}c_{cm} - c_{kb}c_{km}c_{am} - c_{kd}c_{km}c_{cm}$. Adding up all separate elements above, we obtain the change in the square mobility term due to one application of a square move

$$\begin{aligned}\Delta\Big[\frac{1}{2}\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}\boldsymbol{c}^\dagger)\Big] = \quad & 2\left(c_{kd}c_{km}c_{am} + c_{kb}c_{km}c_{cm} - c_{kb}c_{km}c_{am} - c_{kd}c_{km}c_{cm}\right) \\ & +2(k_d^{\mathrm{out}} + k_b^{\mathrm{out}} + k_a^{\mathrm{in}} + k_c^{\mathrm{in}}) - 4\left(c_{id}c_{ib} + c_{am}c_{cm} + 1\right) \qquad \text{(D.1.5)}\end{aligned}$$

- Term 2:

$$\begin{aligned}\Delta\Big[\sum_{ij} k_i^{\mathrm{in}}c_{ij}k_j^{\mathrm{out}}\Big] = \quad & \textstyle\sum_{ij} k_i^{\mathrm{in}}\left(x_{ij} - c_{ij}\right)k_j^{\mathrm{out}} = \sum_{ij} k_i^{\mathrm{in}}\left[\delta_{ia}\delta_{jd} + \delta_{ic}\delta_{jb} - \delta_{ia}\delta_{jb} - \delta_{ic}\delta_{jd}\right]k_j^{\mathrm{out}} \\ = \quad & k_a^{\mathrm{in}}k_d^{\mathrm{out}} + k_c^{\mathrm{in}}k_b^{\mathrm{out}} - k_a^{\mathrm{in}}k_b^{\mathrm{out}} - k_c^{\mathrm{in}}k_d^{\mathrm{out}} \qquad \text{(D.1.6)}\end{aligned}$$

- Term 3:

$$\begin{aligned}\mathrm{Tr}(\mathbf{x}\mathbf{x}^\dagger\mathbf{x}) - \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c}) &= \sum_{ijk}\left[c_{ij} + \Delta_{ij}\right]\left[c_{kj} + \Delta_{kj}\right]\left[c_{ki} + \Delta_{ki}\right] - c_{ij}c_{kj}c_{ki} \qquad \text{(D.1.7)} \\ &= \sum_{ijk}\Big[\Delta_{kj}c_{ij}c_{ki} + \Delta_{ij}c_{kj}c_{ki} + \Delta_{ki}c_{ij}c_{kj} \\ &\quad + \Delta_{ij}\Delta_{kj}c_{ki} + \Delta_{kj}\Delta_{ki}c_{ij} + \Delta_{ij}\Delta_{ki}c_{kj} + \Delta_{ij}\Delta_{kj}\Delta_{ki}\Big]\end{aligned}$$

The product of two Δ terms gives

$$\Delta_{ij}\Delta_{kj} = \delta_{ik}\left(\delta_{jd}(\delta_{ia} - \delta_{ic}) + \delta_{jb}(\delta_{ic} - \delta_{ia})\right) - \delta_{jd}(\delta_{ia}\delta_{kc} + \delta_{ic}\delta_{ka}) - \delta_{jb}(\delta_{ic}\delta_{ka} + \delta_{ia}\delta_{kc})$$

but $\Delta_{ij}\Delta_{ki} = 0$, and in a straightforward way we obtain

$$\begin{aligned}\sum_{ijk}\Delta_{kj}c_{ij}c_{ki} &= \sum_{ijk}\left[\delta_{ka}\delta_{jd} + \delta_{kc}\delta_{jb} - \delta_{ka}\delta_{jb} - \delta_{kc}\delta_{jd}\right]c_{ij}c_{ki} \qquad \text{(D.1.8)} \\ &= \sum_{i}\left[c_{ai}c_{id} + c_{ci}c_{ib} - c_{ai}c_{ib} - c_{ci}c_{id}\right] \qquad \text{(D.1.9)}\end{aligned}$$

Assembling all terms and their symmetric equivalents leads to an expression which can be summarised as

$$\Delta\Big[\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^\dagger\boldsymbol{c})\Big] = \mathrm{MutN}(a,d) + \mathrm{MutN}(c,b) - \mathrm{MutN}(a,b) - \mathrm{MutN}(c,d) - 2(c_{bd} + c_{db} + c_{ac} + c_{ca}) \qquad \text{(D.1.10)}$$

where

$$\mathrm{MutN}(\alpha,\beta) = \sum_i\left[c_{\alpha i}c_{i\beta} + c_{\alpha i}c_{\beta i} + c_{i\alpha}c_{i\beta}\right] \qquad \text{(D.1.11)}$$

- Term 5:

$$\Delta\left[\mathrm{Tr}(\boldsymbol{c}^2)\right] = \mathrm{Tr}(\boldsymbol{x}^2) - \mathrm{Tr}(\boldsymbol{c}^2) = 2(c_{da} + c_{bc} - c_{ba} - c_{dc}) \tag{D.1.12}$$

- Terms 4 and 6:

  The two terms $\frac{1}{2}N^2\langle k\rangle^2$ and $\sum_i k_i^{\mathrm{out}} k_i^{\mathrm{in}}$ do not change, since our stochastic process conserves all degrees.

In combination, the above ingredients lead us to the following update formula for the square mobility (6.1.9), as a result of the edge swap (D.1.1)

$$\begin{aligned}\Delta n_{\square} =& 2(k_d^{\mathrm{out}} + k_b^{\mathrm{out}} + k_a^{\mathrm{in}} + k_c^{\mathrm{in}}) + 2\left(c_{kd}c_{km}c_{am} + c_{kb}c_{km}c_{cm} - c_{kb}c_{km}c_{am} - c_{kd}c_{km}c_{cm}\right) \\ & - 4\left(c_{id}c_{ib} + c_{am}c_{cm} + 1\right) - [k_a^{\mathrm{in}}k_d^{\mathrm{out}} + k_c^{\mathrm{in}}k_b^{\mathrm{out}} - k_a^{\mathrm{in}}k_b^{\mathrm{out}} - k_c^{\mathrm{in}}k_d^{\mathrm{out}}] \\ & + \mathrm{MutN}(a,d) + \mathrm{MutN}(c,b) - \mathrm{MutN}(a,b) - \mathrm{MutN}(c,d) - 2(c_{bd} + c_{db} + c_{ac} + c_{ca}) \\ & + c_{da} + c_{bc} - c_{ba} - c_{dc}\end{aligned} \tag{D.1.13}$$

D.1.2 Change in $n_{\triangle}(c)$ following one square-type move

The different terms in the triangle mobility term (to be called Term 7, Term 8, Term 9 and Term 10, to avoid confusion with the previous section) are

$$n(\boldsymbol{c})_{\triangle} = \quad \tfrac{1}{3}\mathrm{Tr}(\mathbf{c}^3) - \mathrm{Tr}(\mathbf{c}^{\updownarrow}\mathbf{c}^2) + \mathrm{Tr}(\mathbf{c}^{\updownarrow 2}\mathbf{c}) - \tfrac{1}{3}\mathrm{Tr}(\mathbf{c}^{\updownarrow 3}) \tag{D.1.14}$$

- Term 7:

$$\Delta\mathrm{Tr}(\mathbf{c}^3) = \sum_{ijk}\left[x_{ij}x_{jk}x_{ki} - c_{ij}c_{jk}c_{ki}\right] = 3\sum_i (c_{ia}c_{di} + c_{ic}c_{bi} - c_{ia}c_{bi} - c_{ic}c_{di}) \tag{D.1.15}$$

- Term 8:

  Here we have to inspect first how the matrix $\boldsymbol{c}^{\updownarrow}$ of double bonds is affected by a square move:

$$x_{ij}^{\updownarrow} = c_{ij}^{\updownarrow} + \Delta_{ij}^{\updownarrow} + \Delta_{ji}^{\updownarrow} \tag{D.1.16}$$

  with

$$\Delta_{ij}^{\updownarrow} = \delta_{ia}\delta_{jd}c_{da} + \delta_{ic}\delta_{jb}c_{bc} - \delta_{ia}\delta_{jb}c_{ba} - \delta_{ic}\delta_{jd}c_{dc} \tag{D.1.17}$$

It follows that

$$\Delta\mathrm{Tr}(\mathbf{c}^{\updownarrow}\mathbf{c}^2) = \mathrm{Tr}(\mathbf{x}^{\updownarrow}\mathbf{x}^2) - \mathrm{Tr}(\mathbf{c}^{\updownarrow}\mathbf{c}^2) = \sum_{ijk}\left(c^{\updownarrow}_{ij} + \Delta^{\updownarrow}_{ij} + \Delta^{\updownarrow}_{ji}\right)\left(c_{jk} + \Delta_{jk}\right)(c_{ki} + \Delta_{ki}) - \mathrm{Tr}(\mathbf{c}^{\updownarrow}\mathbf{c}^2)$$

Arguments similar to those employed before show that $\sum_j \Delta^{\updownarrow}_{ij}\Delta_{jk} = \sum_i \Delta^{\updownarrow}_{ij}\Delta_{ki} = 0$, whereas the remaining two 'compound' terms give

$$\begin{aligned}\sum_{ijk}\Delta^{\updownarrow}_{ji}\Delta_{jk}c_{ki} =& \sum_{ijk}\left(\delta_{ja}\delta_{id}c_{da} + \delta_{jc}\delta_{ib}c_{bc} - \delta_{ja}\delta_{ib}c_{ba} - \delta_{jc}\delta_{id}c_{dc}\right) \\ &\times\left(\delta_{ja}\delta_{kd} + \delta_{jc}\delta_{kb} - \delta_{ja}\delta_{kb} - \delta_{jc}\delta_{kd}\right)c_{ki} \\ =&(c_{da} + c_{dc})\delta_{id}(\delta_{kd} - \delta_{kb})c_{ki} + (c_{bc} + c_{ba})\delta_{ib}(\delta_{kb} - \delta_{kd})c_{ki} \\ =&-(c_{da} + c_{dc})c_{bd} - (c_{bc} + c_{ba})c_{db}\end{aligned} \tag{D.1.18}$$

and

$$\begin{aligned}\sum_{ijk}\Delta^{\updownarrow}_{ji}c_{jk}\Delta_{ki} =&(c_{da}+c_{ba})\delta_{ja}(\delta_{ka}-\delta_{kc})c_{jk} + (c_{bc}+c_{dc})\delta_{jc}(\delta_{kc}-\delta_{ka})c_{jk} \\ =&-\left[(c_{da} + c_{ba})c_{ac} + (c_{bc} + c_{dc})c_{ca}\right]\end{aligned} \tag{D.1.19}$$

The product of three Deltas can be immediately seen to be zero by earlier arguments (repeated suffix in different positions). The other terms evaluate as follows:

$$\begin{aligned}\sum_{ijk}\Delta^{\updownarrow}_{ij}c_{jk}c_{ki} &= \sum_{\beta\in\{a,c\},\ \alpha\in\{d,b\}}\mathbb{I}(\alpha,\beta)c_{\alpha k}c_{\alpha\beta}c_{k\beta} \\ \sum_{ijk}\Delta_{ki}c^{\updownarrow}_{ij}c_{jk} &= \sum_{\beta\in\{a,c\},\ \alpha\in\{d,b\}}\mathbb{I}(\alpha,\beta)c^{\updownarrow}_{\alpha k}c_{k\beta} \\ \sum_{ijk}\Delta_{jk}c^{\updownarrow}_{ij}c_{ki} &= \sum_{\beta\in\{a,c\},\ \alpha\in\{d,b\}}\mathbb{I}(\alpha,\beta)c_{\alpha k}c^{\updownarrow}_{k\beta}\end{aligned} \tag{D.1.20}$$

where $\mathbb{I}(\alpha,\beta)$ is an indicator function which evaluates to 1 if bond (α,β) is created by the present move, to -1 if the bond (α,β) is destroyed, and zero otherwise. Similarly

$$\begin{aligned}\sum_{ijk}\Delta^{\updownarrow}_{ji}c_{jk}c_{ki} &= \sum_{ijk}c_{jk}c_{ki}\left(\delta_{ja}\delta_{id}c_{da} + \delta_{jc}\delta_{ib}c_{bc} - \delta_{ja}\delta_{ib}c_{ba} - \delta_{jc}\delta_{id}c_{dc}\right) \\ &= \sum_{k}\left(c_{ak}c_{kd}c_{da} + c_{ck}c_{kb}c_{bc} - c_{ak}c_{kb}c_{ba} - c_{ck}c_{kd}c_{dc}\right)\end{aligned}$$

Putting all of these sub-terms together yields:

$$\begin{aligned}\Delta\left[\mathrm{Tr}(\mathbf{c}^{\updownarrow}\mathbf{c}^2)\right] = &\sum_{\beta\in\{a,c\},\ \alpha\in\{d,b\}}\mathbb{I}(\alpha,\beta)\sum_k\left[c_{\alpha\beta}(c_{\alpha k}c_{k\beta} + c_{\beta k}c_{k\alpha}) + c^{\updownarrow}_{\alpha k}c_{k\beta} + c_{\alpha k}c^{\updownarrow}_{k\beta}\right] \\ &- c_{bd}\left(c_{da} + c_{dc}\right) + c_{ac}\left(c_{da} + c_{ba}\right) + c_{db}\left(c_{bc} + c_{ba}\right) + c_{ca}\left(c_{bc} + c_{dc}\right)\end{aligned} \tag{D.1.21}$$

- Terms 9 and 10:

  The same steps as followed to calculate term 8 can be also be applied to terms 9 and 10, In combination, the above ingredients lead us to the following update formula for the triangle mobility (6.1.10), as a result of the edge swap (D.1.1):

$$\begin{aligned}\Delta n_{\triangle} = \sum_{\beta\in\{1,3\},\ \alpha\in\{4,2\}} \mathbb{I}(\alpha,\beta) \sum_{k} \Big[& c_{\alpha k}c_{k\beta} - c_{\alpha\beta}(c_{\alpha k}c_{k\beta} + c_{\beta k}c_{k\alpha}) \\ & - c^{\updownarrow}_{\alpha k}c_{k\beta} - c_{\alpha k}c^{\updownarrow}_{k\beta} + c^{\updownarrow}_{\alpha k}c^{\updownarrow}_{k\beta} - c_{\alpha\beta}\left(c^{\updownarrow}_{\alpha k}c^{\updownarrow}_{k\beta} + c^{\updownarrow}_{k\alpha}c^{\updownarrow}_{\beta k}\right) + c_{\alpha\beta}\left(c^{\updownarrow}_{\alpha k}c_{k\beta} + c_{\alpha k}c^{\updownarrow}_{k\beta} + c^{\updownarrow}_{k\alpha}c_{\beta k} + c_{k\alpha}c^{\updownarrow}_{\beta k}\right)\Big] \\ & - c_{bd}(c_{db}-1)\left(c_{da}(1-c_{ba}) + c_{dc}(1-c_{bc})\right) - c_{ac}(c_{ca}-1)\left(c_{da}(1-c_{dc}) + c_{ba}(1-c_{bc})\right) \\ & - c_{db}(c_{bd}-1)\left(c_{bc}(1-c_{dc}) + c_{ba}(1-c_{da})\right) - c_{ca}(c_{ac}-1)\left(c_{bc}(1-c_{ba}) + c_{dc}(1-c_{da})\right)\end{aligned} \tag{D.1.22}$$

D.1.3 Change in $n_{\square}(\boldsymbol{c})$ following one triangle-type move

The triangle move is a transformation from network $\boldsymbol{c}$ to network $\mathbf{x}$, characterised by $x_{ij} = c_{ij} + \Omega_{ij}$ with

$$\Omega_{ij} \quad = \quad \delta_{ib}\delta_{ja} + \delta_{ic}\delta_{jb} + \delta_{ia}\delta_{jc} - \delta_{ia}\delta_{jb} - \delta_{ib}\delta_{jc} - \delta_{ic}\delta_{ja} \tag{D.1.23}$$

The terms which make up the square mobility term are

$$n_{\square}(\boldsymbol{c}) = \quad \tfrac{1}{2}\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) - \textstyle\sum_{ij} k^{\mathrm{in}}_i c_{ij} k^{\mathrm{out}}_j + \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) + \tfrac{1}{2}N^2\langle k\rangle^2 + \tfrac{1}{2}\mathrm{Tr}(\boldsymbol{c}^2) - \textstyle\sum_i k^{\mathrm{out}}_i k^{\mathrm{in}}_i$$

- Term 2:

$$\sum_{ij} k^{\mathrm{in}}_i \Omega_{ij} k^{\mathrm{out}}_j = \sum_{\alpha,\beta\in\{1,2,3\}} \mathbb{I}(\alpha,\beta) k^{\mathrm{in}}_{\alpha} k^{\mathrm{out}}_{\beta} \tag{D.1.24}$$

- Term 3:

$$\Delta\Big[\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c})\Big] = \mathrm{Tr}(\mathbf{x}\mathbf{x}^{\dagger}\mathbf{x}) - \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) = \sum_{ijk}\Big(c_{ij}+\Omega_{ij}\Big)\Big(c_{kj}+\Omega_{kj}\Big)(c_{ki}+\Omega_{ki}) - c_{ij}c_{kj}c_{ki}$$

  We consider each subterm separately:

$$\begin{aligned}\sum_{ijk}\Omega_{ij}c_{kj}c_{ki} \quad &= \quad \sum_{ijk}\Omega_{ki}c_{kj}c_{ij} = 0 \\ \sum_{ijk}\Omega_{kj}c_{ij}c_{ki} \quad &= \quad \sum_{\alpha,\beta\in\{1,2,3\}} \mathbb{I}(\alpha,\beta)\sum_{i} c_{\alpha i}c_{i\beta}\end{aligned} \tag{D.1.25}$$

Clearly

$$\sum_{ijk} \Omega_{ij}\Omega_{kj} = \sum_{ijk} \Omega_{ki}\Omega_{kj} = 0$$

since the Ω kills any suffix repeated in the same position. Furthermore,

$$\sum_{i} \Omega_{ij}\Omega_{ki} = \sum_{\alpha,\beta\in\{a,b,c\}} \left(1-\delta_{\alpha\beta}\right)\delta_{j\alpha}\delta_{k\beta} - 2\sum_{\alpha\in\{a,b,c\}} \delta_{j\alpha}\delta_{k\alpha} \tag{D.1.26}$$

hence

$$\sum_{ijk} \Omega_{ij}\Omega_{ki}c_{kj} = 3 \tag{D.1.27}$$

So it follows that

$$\mathrm{Tr}(\mathbf{x}\mathbf{x}^{\dagger}\mathbf{x}) - \mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}) = 3 + \sum_{\alpha,\beta\in\{a,b,c\}} \mathbb{I}(\alpha,\beta)\sum_{i} c_{\alpha i}c_{i\beta} \tag{D.1.28}$$

- Term 5:

  We observe that $\sum_{ij} c_{ji}\Omega_{ij} = 3$ and $\sum_{ij} \Omega_{ij}\Omega_{ji} = -6$. We conclude that $\Delta[\mathrm{Tr}(\boldsymbol{c}^2)] = 0$. This is as expected, since double bonds cannot participate in a *triangle swap*.

- Term 1:

  Finally we return to Term 1 using the various shortcuts derived above. We recall that a suffix repeated in the same position sends the term to zero. Hence, we already know that all terms featuring the product of 3 or 4 Ω terms will be zero. Next

$$\sum_{j} \Omega_{ij}c_{kj} = \sum_{\alpha,\beta\in\{1,2,3\}} \mathbb{I}(\alpha,\beta)\delta_{i\alpha}c_{k\beta} \tag{D.1.29}$$

  From this it follows that

$$\sum_{ij} \Omega_{ij}c_{kj}\Omega_{km}c_{im} = 0 \tag{D.1.30}$$

  Finally,

$$\sum_{ijkm} \Omega_{ij}c_{kj}c_{km}c_{im} \quad = \quad c_{km}\left[c_{k1}\left(c_{2m}-c_{3m}\right) + c_{k2}\left(c_{3m}-c_{1m}\right) + c_{k3}\left(c_{1m}-c_{2m}\right)\right] \tag{D.1.31}$$

  (and similarly with the other terms related to this one by simple permutations). Overall we thus find

$$\Delta\mathrm{Tr}(\boldsymbol{c}\boldsymbol{c}^{\dagger}\boldsymbol{c}\boldsymbol{c}^{\dagger}) = 4\sum_{km} c_{km}\sum_{\alpha,\beta\in\{a,b,c\}} \mathbb{I}(\alpha,\beta)c_{\alpha m}c_{k\beta} \tag{D.1.32}$$

Collecting all these terms together, we see that the expected change in the square mobility term after the application of a single triangle type move is

$$\Delta\left[n_{\square}\right] = \sum_{\alpha,\beta\in\{a,b,c\}} \mathbb{I}(\alpha,\beta)\Big[c_{\alpha i}c_{i\beta} + k_{\beta}^{\text{out}}k_{\alpha}^{\text{in}} + 2\sum_{km} c_{km}c_{\alpha m}c_{k\beta}\Big] + 3 \tag{D.1.33}$$

D.1.4 Change in $n_{\triangle}(\boldsymbol{c})$ following one triangle-type move

This final incremental term is best evaluated by an algorithm which, for each edge created or destroyed, searches for mono-directed triangles that have been created or destroyed.

APPENDIX E

DATASETS USED IN THIS THESIS

Dataset	Species	Experimental Method
Titz et al. [2008]	*T. pallidum*	Y2H
LaCount et al. [2005]	*P. falciparum*	Y2H
Ewing et al. [2007]	*H. sapiens*	AP-MS
Rual et al. [2005]	*H. sapiens*	Y2H
Arifuzzaman et al. [2006]	*E. coli*	AP-MS
Collins et al. [2007a]	*S. cerevisiae*	AP-MS
Ito et al. [2001]	*S. cerevisiae*	Y2H
Krogan et al. [2006]	*S. cerevisiae*	AP-MS
Tarassov et al. [2008]	*S. cerevisiae*	PCA
Uetz et al. [2000]	*S. cerevisiae*	Y2H
Von Mering et al. [2002]	*S. cerevisiae*	DI

Table E.1: A list of protein-protein interaction databases used, based on Fernandes et al. [2010]. Y2H refers to yeast two hybrid; AP-MS refers to affinity purification mass spectroscopy; PCA refers to protein complementation assay; and, DI refers to data integration.

APPENDIX F

COMPUTER CODE WRITTEN FOR THIS PROJECT

Several programs have been created in C++ for the purpose of illustrating the theory described, or testing out ideas for applications. Some key subroutines are transcribed below in a truncated form.

The code is based on two classes: CConnectionMatrix and CCoordinate. CCoordinate defines an ordered pair ('co-ordinate') (x, y) and operations allowed on 'coordinates'. The code will go on to use these co-ordinates to represent a link running between x and y. CConnectionMatrix class defines a 'connection matrix'-type object and its associated 'member functions' which record properties of the CConnectionMatrix object (e.g. lists of neighbours, degree degree correlation matrix, degree distribution). In the Visual Studio development environment one can benefit from being able to tab through the code and watch the member functions being updated, which is helpful for debugging and for general understanding. The final step is to define 'public' functions on the class. The purpose of these is to pre-empt the questions that the main code will wish to ask about the CConnectionMatrix object, and to define functions to allow you to 'ask' these questions in a straighforward way, with natural inputs and outputs. For example, you can ask AreNeighbours(x,y). The most time consuming part of this is the

Initialise step. After this, the code increments the member functions are required. Front loading the work will substantially improve the efficiency of the code for larger networks. The code starts in Console.cpp, and request user input: filename, option etc. It then immediately moves into Main.cpp. This is to allow maximum flexibility if we wanted to update the interface at a later date.

In our code we use object-oriented design. The idea is to have a workbench of a code, so we can develop new related software using high-level predefined commands and properties on our network objects. While it initially looks complicated as a whole, the benefit is that within Main.cpp the code becomes descriptive, with a lot of natural-type language. Although it has been a work in progress to develop conventions to achieve this aim, it is for example relatively easy to toggle to be able to load between different data formats.

F.1 Directed randomiser code

F.1.1 Coordinate.h and Coordinate.cpp

```
// Basic class designed to represent a co-ordinate (x, y) where x and y are integers
class CCoordinate {
public:
    CCoordinate() : m_x(0), m_y(0){}                                  // default contructor
    CCoordinate(const int& x, const int& y) : m_x(x), m_y(y){}        // constructor
    CCoordinate(const CCoordinate& x) : m_x(x.m_x), m_y(x.m_y){}      // copy constructor
    virtual ~CCoordinate(){}                                          // destructor
    bool operator==(const CCoordinate& x) const {                     // equality operator
        return (x.m_x == m_x) && (x.m_y == m_y); }
    int x() const { return m_x; }                                     // accessor x
    int y() const { return m_y; }                                     // accessor y
    void put_x(int x) { m_x = x; }                                    // mutator x
    void put_y(int y) { m_y = y; }                                    // mutator
    void shufflelink() { int tmp = m_x; m_x = m_y; m_y = tmp; }       // (x,y) -> (y,x)
private:
    int m_x, m_y; };
```

F.1.2 ConnectionMatrix.h and ConnectionMatrix.cpp

```
enum InitStage { None = 0, Base, DirectedGraph };
enum SwapType { NoSwap = 0, Triangle, Square };
enum Direction { In = 0, Out };

// CConnectionMatrix class defined to allow easy workings with the connection matrix
class CConnectionMatrix
{
public:
    CConnectionMatrix(const char*);            // Constuctor (from file)
    void InitialiseDirectedMobilityTerms();
    int NumberSwitches();
    CCoordinate& RandomLink();
    CCoordinate& RandomUnconnectedLink(const CCoordinate&, SwapType&);
    bool AcceptRejectDirected(const CCoordinate&, const CCoordinate&, SwapType);
    void SwitchPair(CCoordinate&, CCoordinate&);
    void SwitchPairs(int);
    void ReverseTriangle(CCoordinate&, CCoordinate&);
    void SaveTab(int);
    double HammingDistance(CConnectionMatrix&);
    const unsigned int& NumberSquares() const {return m_Squares;}
    const unsigned int& NumberTriangles() const {return m_Triangles;}
private:
    // Private functions. Not to be used outside the class
    void Initialise();
    void MobilityTermSquare();
    void MobilityTermTriangle();
    vector<int> GetNeighbours(int, Direction) const;
    int IncrementalMobilityTermsSquare(int, int, int, int, SwapType) const;
    int IncrementalMobilityTermsTriangle(int, int, int, int, SwapType) const;
    int CCTC_TrianglesBasedOn(int, int) const;
    bool AreNeighbours(int i, int j) const { return m_Connection[ (i - 1) * m_Nodes + j - 1] == '1';}
    bool WereNeighbours(int i, int j) const { return m_OrigConnection[ (i - 1) * m_Nodes + j - 1] == '1';}
private:
    vector<char> m_OrigConnection, m_Connection;        // Original and mutated connections
```

```
    unsigned int m_Nodes, m_Edges;                      // Number of nodes and connections
    vector<CCoordinate> m_Neighbours;                   // List of all connected pairs
    InitStage m_InitStage;                              // Initialisation stage
    double m_DegreeAvg, m_DegreeVar;                    // Average Degree, variance
    vector<unsigned int> m_Degrees_out, m_Degrees_in;   // Degrees out/in
    unsigned int m_MaxDegree_out, m_MaxDegree_in;       // Maximum out/it degree
    vector<double> m_DegreePDF_out, m_DegreePDF_in;     // Probability density function of the degrees
    unsigned int m_Squares, m_Triangles;                // Number of triangles/squares in the network
    vector<string> m_name_list; };                      // stores the 'key' to decode which number corresponds to which node.

CConnectionMatrix::CConnectionMatrix(const char* filename) : m_InitStage(None) {
    FILE* file;
    fopen_s(&file, filename, "r");
    if(file == NULL)
        throw "No file loaded!";        // Check that the file has been opened correctly
    char dummy[200];
    string orf1, orf2;
    int i = 0, counter = 0;
    int orf1_number = -1, orf2_number = -1;
    while ( !feof(file) ) {             // Do the loop until we reach the end of the file
        char temp_dummy =' ' ;
        int counter2 = 0;               // Read in word by word, terminate when I see a tab or a line-end
        while( temp_dummy!='\t' && temp_dummy!='\n') {
            dummy[counter2] = temp_dummy;
            fscanf_s(file, "%c", &temp_dummy);
            if(temp_dummy == ' ')
                break;
            ++counter2; }
        if(temp_dummy == ' ')
            break;
        dummy[counter2] = '\0';
        if(counter % 2 == 0 ) {          // Check if we're on name list (but don't bother if we're on first pass)
            if(counter == 0 ) {
                m_name_list.push_back(dummy);
                orf1_number = 0; }
            else {
                for(unsigned int i = 0; i < m_name_list.size(); ++i)
                    if( m_name_list[i] == dummy)
                        orf1_number = i;
                if(orf1_number == -1) {
                    orf1_number =  m_name_list.size();
                    m_name_list.push_back(dummy); } } }
        if(counter % 2 == 1 ) {
            for(unsigned int i = 0; i < m_name_list.size(); ++i)
                if( m_name_list[i] == dummy)
                    orf2_number = i;
            if(orf2_number == -1) {
                orf2_number =  m_name_list.size();
                m_name_list.push_back(dummy); }
            if(orf1_number != orf2_number) {
                CCoordinate coord(orf1_number + 1, orf2_number + 1);
                m_Neighbours.push_back(coord); }
            orf1_number = -1;
            orf2_number = -1; }
        ++counter; }
    m_Connection.resize( m_name_list.size() * m_name_list.size(), '0');
    m_Nodes= m_name_list.size();
    for(unsigned int i = 0; i<m_Neighbours.size(); i++) {   // Read through neighbours vector and populate m_connection
int x = m_Neighbours[i].x() - 1;
int y = m_Neighbours[i].y() - 1;
        m_Connection[x * m_Nodes + y] = '1'; }
```

```
    fclose(file);                       // Close file
    m_Edges = m_Neighbours.size();      // Count edges
    m_InitStage = Base;                 // Step to 'base'
    Initialise(); }                     // Now initialise

void CConnectionMatrix::Initialise() {
    if (m_InitStage == None)
        throw "Cannot initialise from NONE InitStage";
    // Calculate the 'out' statistics
    vector<int> *tempNeighbours = new vector<int> [m_Nodes];
    m_Degrees_out.resize(m_Nodes);
    for(unsigned int i = 0; i < m_Nodes; ++i) {
        tempNeighbours[i] = GetNeighbours(i + 1, Out);
        m_Degrees_out[i] = tempNeighbours[i].size();
        m_MaxDegree_out = max(m_Degrees_out[i], m_MaxDegree_out); }
    m_DegreePDF_out.resize(m_MaxDegree_out + 1, 0.0);
    for(unsigned int i = 0; i < m_Nodes; ++i) {
        ++(m_DegreePDF_out[m_Degrees_out[i]]);
        m_DegreeAvg += m_Degrees_out[i];
        m_DegreeVar += m_Degrees_out[i] * m_Degrees_out[i];}
    for(unsigned int i = 0; i <= m_MaxDegree_out; ++i)
        m_DegreePDF_out[i] /= m_Nodes;
    m_DegreeAvg /= m_Nodes;
    m_DegreeVar /= m_Nodes;
    delete[] tempNeighbours;
    // And now the 'in' statistics
    vector<int> *tempNeighbours_in = new vector<int> [m_Nodes];
    m_Degrees_in.resize(m_Nodes);
    for(unsigned int i = 0; i < m_Nodes; ++i) {
        tempNeighbours_in[i] = GetNeighbours(i + 1, In);
        m_Degrees_in[i] = tempNeighbours_in[i].size();
        m_MaxDegree_in = max(m_Degrees_in[i], m_MaxDegree_in); }
    m_DegreePDF_in.resize(m_MaxDegree_in + 1, 0.0);
    for(unsigned int i = 0; i < m_Nodes; ++i)
        ++(m_DegreePDF_in[m_Degrees_in[i]]);
    for(unsigned int i = 0; i <= m_MaxDegree_in; ++i)
        m_DegreePDF_in[i] /= m_Nodes;
    delete[] tempNeighbours_in;
    m_InitStage = DirectedGraph; }

vector<int> CConnectionMatrix::GetNeighbours(int i, Direction direction) const {
    vector<int> neighbours(0);
    switch(direction) {
    case In:
        for (unsigned int j = 1; j <= m_Nodes; ++j)
            if (AreNeighbours(j, i) == true && i != j)
                neighbours.push_back(j);
        break;
    case Out :
        for (unsigned int j = 1; j <= m_Nodes; ++j)
            if (AreNeighbours(i, j) == true && i != j)
                neighbours.push_back(j);
        break; }
return neighbours; }

CCoordinate& CConnectionMatrix::RandomLink(void) {
    MTRand mtrand2;
    int random = mtrand2.randInt( m_Neighbours.size() -1 );
    CCoordinate& retNeighbour = m_Neighbours[random];
    return retNeighbour; }
```

```
CCoordinate& CConnectionMatrix::RandomUnconnectedLink(const CCoordinate& link1, SwapType& swapType) {
    int randPosition;
    // Recall, for directed case, I need a proportion of triangle and square moves.
    // Two edges fully define a proposed triangle or square move
    // Proportion of valid square to traingle moves will be the same as ratio of total number
    // or square or triangle moves
    // No pair of edges can form part of both types of moves.
    // Hence here I check if my two edges define either move, and execute the move if i can
    // Due to the above, this will automatically give me the right proportion of moves attempted
    // I have 1 problem: not all link1's will have a valid swap partner. So I need to insert a break clause for that possibility.
    MTRand mtrand3;
    randPosition = mtrand3.randInt( m_Neighbours.size() -1 );
    if( link1.x() == m_Neighbours[randPosition].y()) {      // First check if I have a triangle one way
        if( AreNeighbours(link1.y(), link1.x()) == false &&
            AreNeighbours(m_Neighbours[randPosition].y(), m_Neighbours[randPosition].x() ) == false &&
            AreNeighbours(m_Neighbours[randPosition].x(), link1.y()) == false ) {
                if( AreNeighbours(link1.y(), m_Neighbours[randPosition].x()) )
                    swapType = Triangle; } }
    if( link1.y() == m_Neighbours[randPosition].x()) {
        if( AreNeighbours(link1.y(), link1.x()) == false &&
            AreNeighbours(m_Neighbours[randPosition].y(), m_Neighbours[randPosition].x()) == false &&
            AreNeighbours(link1.x(), m_Neighbours[randPosition].y()) == false) {
                if( AreNeighbours(m_Neighbours[randPosition].y(), link1.y() ) )
                    swapType = Triangle; } }
    if( link1.x()!= m_Neighbours[randPosition].x() &&
        link1.x()!= m_Neighbours[randPosition].y() &&
        link1.y()!= m_Neighbours[randPosition].x() &&
        link1.y()!= m_Neighbours[randPosition].y() &&
        AreNeighbours(m_Neighbours[randPosition].x(), link1.y()) == false &&
        AreNeighbours(link1.x(), m_Neighbours[randPosition].y()) == false)
        swapType = Square;
    return m_Neighbours[randPosition]; }

void CConnectionMatrix::SwitchPair(CCoordinate& link1, CCoordinate& link2) {
    int y1 = link1.y(), y2 = link2.y();
int x1 = link1.x(), x2 = link2.x();
    link1.put_y(y2); link2.put_y(y1);
m_Connection[(y1-1) + m_Nodes * (x1-1)] = false;
m_Connection[(y2-1) + m_Nodes * (x2-1)] = false;
m_Connection[(y1-1) + m_Nodes * (x2-1)] = true;
m_Connection[(y2-1) + m_Nodes * (x1-1)] = true; }

void CConnectionMatrix::SwitchPairs(int num) {
    int i(0);
    InitialiseDirectedMobilityTerms();
    FILE *file;
    fopen_s( &file, "C:/data/directed_randomiser/mobility_terms_accept_all.txt", "a");
    int obsinterval = max(1, num / 100);
    fprintf(file," I'm attempting %d switches Outputting every %d steps \n", num, obsinterval);
    double mob_total1000_square = 0.0;
    double mob_total1000_triangle = 0.0;
    for (i = 0; i < num; i++) {
        SwapType swapType(NoSwap);
        while(swapType == NoSwap) {
            CCoordinate& link1 = RandomLink();
            CCoordinate& link2 = RandomUnconnectedLink(link1, swapType);
            if(swapType == Triangle) {
                double randNum = static_cast<double>(rand()) / static_cast<double>(RAND_MAX);
                double correction = 1 / 3;
                if(randNum > correction)
                    swapType = NoSwap; }
```

```
            if(swapType != NoSwap) {
                m_Squares += IncrementalMobilityTermsSquare(link1.x(), link1.y(),link2.x(),  link2.y(), swapType);
                m_Triangles += IncrementalMobilityTermsTriangle(link1.x(), link1.y(),link2.x(),  link2.y(), swapType);
                if(i % obsinterval == 0 ) {
                    fprintf(file," %f \t %f \n", mob_total1000_square/obsinterval, mob_total1000_triangle/obsinterval);
                    mob_total1000_square = m_Squares;
                    mob_total1000_triangle = m_Triangles; }
                else {
                    mob_total1000_square += m_Squares;
                    mob_total1000_triangle += m_Triangles; }
            }
            if(swapType == Square)
                SwitchPair(link1, link2);
            if(swapType == Triangle)
                ReverseTriangle(link1, link2); } } }

double CConnectionMatrix::HammingDistance(CConnectionMatrix& rhs) {
    unsigned int i(0);
    double h(0.0);
    unsigned int size1 = m_Connection.size(), size2 = rhs.m_Connection.size();
    if (size1 != size2)
        throw "Not the same size.";
    for (i = 0; i < size1; i++)
        if (m_Connection[i] != rhs.m_Connection[i])
            h++;
    return h / (2.0 * ((double) m_Edges)); }

void CConnectionMatrix::SaveTab(int choice) {
    FILE *file;
    if(choice == 13)
        fopen_s( &file, "C:/data/directed_randomiser/Output_network_correct_dynamics.txt", "w");
    if(choice == 14)
        fopen_s( &file, "C:/data/directed_randomiser/Output_network_accept_all.txt", "w");
    if(choice == 15)
        fopen_s( &file, "C:/data/directed_randomiser/target_pi.txt", "w");
    if(choice == 16)
        fopen_s( &file, "C:/data/directed_randomiser/Output_network_debugger.txt", "w");
    for(unsigned int k = 0; k < m_Neighbours.size(); ++k)
        fprintf(file,"%d\t%d\n", m_Neighbours[k].x() , m_Neighbours[k].y());
    fclose(file);
    FILE *file2;
    fopen_s( &file2, "c:/data/directed_randomiser/descriptive_Output_network.txt", "w");
    for(unsigned int j = 0; j < m_Neighbours.size(); ++j) {
        int source_ref = m_Neighbours[j].x();
        source_ref -= 1;
        string source = m_name_list[source_ref];
        int sink_ref = m_Neighbours[j].y();
        sink_ref -= 1;
        string sink = m_name_list[sink_ref];
        fprintf(file2,"%s\t%s\n", source.c_str() , sink.c_str()); }
    fprintf(file2,"\n \n \n \n");
    fclose(file2); }

int CConnectionMatrix::NumberSwitches() {
    int kav = (int) 2 * m_Edges / m_Nodes;
    int cycles = 10 *  kav * m_Nodes   + 10 * m_Nodes + 100 * kav + 100 ;
    return cycles; }

void CConnectionMatrix::ReverseTriangle(CCoordinate& link1, CCoordinate& link2) {
    int x1 = link1.x(), y1 = link1.y();
    int x2 = link2.x(), y2 = link2.y();
```

```
    if( y1 != x2) {
if(x1 != y2)    // Sanity check that triangle is a triangle
            throw "Whoah! My triangles aren't triangles!";
        else {
            int counter = 0;
            int position;
            for(unsigned int i = 0; i < m_Neighbours.size(); ++i)
                if(m_Neighbours[i].y() == x2 && m_Neighbours[i].x() == y1) {
                    position = i;
                    ++counter; }
            if(counter != 1)
                throw "Oops - I was supposed to find exactly one link to close the triangle";
            else {
                CCoordinate& link3 = m_Neighbours[position];
                int x3 = link3.x(), y3 = link3.y();
                link1.shufflelink();
                link2.shufflelink();
                link3.shufflelink();
                m_Connection[(y1-1) + m_Nodes * (x1-1)] = false;
                m_Connection[(y2-1) + m_Nodes * (x2-1)] = false;
                m_Connection[(y3-1) + m_Nodes * (x3-1)] = false;
                m_Connection[(x1-1) + m_Nodes * (y1-1)] = true;
                m_Connection[(x2-1) + m_Nodes * (y2-1)] = true;
                m_Connection[(x3-1) + m_Nodes * (y3-1)] = true; } } }
    if(x1 != y2) {
        if( y1 != x2)
            throw "Whoah! My triangles aren't triangles!";
        else {
            int counter = 0;
            int position;
            for(unsigned int i = 0; i< m_Neighbours.size(); i++)
                if(m_Neighbours[i].y()==x1 && m_Neighbours[i].x()==y2  ) {
                    position=i;
                    counter++; }
            if(counter != 1)
                throw "Oops - I was supposed to find exactly one link to close the triangle";
            else {
                CCoordinate& link3 = m_Neighbours[position];
                int x3 = link3.x(), y3 = link3.y();
                link1.shufflelink();
                link2.shufflelink();
                link3.shufflelink();
                m_Connection[(x1-1) + m_Nodes * (y1-1)] = false;
                m_Connection[(x2-1) + m_Nodes * (y2-1)] = false;
                m_Connection[(x3-1) + m_Nodes * (y3-1)] = false;
                m_Connection[(y1-1) + m_Nodes * (x1-1)] = true;
                m_Connection[(y2-1) + m_Nodes * (x2-1)] = true;
                m_Connection[(y3-1) + m_Nodes * (x3-1)] = true; } } } }

int CConnectionMatrix::IncrementalMobilityTermsSquare(int i1, int i2, int i3, int i4, SwapType swapType) const {
    if(swapType == Square) {
        int term1 = -4, term2 = 0, term3 = 0, term5 = 0, temp = 0;
        // Term 1 = -4 c_{k2} c_{k4} -4 c_{1m} c_{3m}
        for(unsigned int i = 0; i < m_Nodes; ++i)
            temp += m_Connection[i*m_Nodes+(i4-1)]*m_Connection[i*m_Nodes+(i2-1)]
                  + m_Connection[(i1-1)*m_Nodes+i]*m_Connection[(i3-1)*m_Nodes+i];
        term1 -= 4*temp;
        temp = 0;
        vector<int> Out_i1 = GetNeighbours(i1, Out), in_i2 = GetNeighbours(i2, In);
        vector<int> Out_i3 = GetNeighbours(i3, Out), in_i4 = GetNeighbours(i4, In);
        // 2(c_{k4}c_{km}c_{1m}+c_{k2}c_{km}c_{3m}-c_{k2}c_{km}c_{1m}-c_{k4}c_{km}c_{3m})
```

```
        for(unsigned int i = 0; i < in_i2.size(); ++i) {
            for(unsigned int j = 0; j < Out_i3.size(); ++j)
                if(AreNeighbours(in_i2[i], Out_i3[j]))
                    ++temp;
            for(unsigned int j = 0; j < Out_i1.size(); ++j)
                if(AreNeighbours(in_i2[i], Out_i1[j]))
                    --temp; }
        for(unsigned int i = 0; i < in_i4.size(); ++i) {
            for(unsigned int j = 0; j < Out_i3.size(); ++j)
                if(AreNeighbours(in_i4[i], Out_i3[j]))
                    --temp;
            for(unsigned int j = 0; j < Out_i1.size(); ++j)
                if(AreNeighbours(in_i4[i], Out_i1[j]))
                    ++temp;}
        term1 += 2*temp;
        // Correction 2(k_{4}^{in}+k_{2}^{in}+k_{1}^{Out}+k_{3}^{Out})
        term1 += 2*(m_Degrees_out[i1-1]+m_Degrees_out[i3-1]+m_Degrees_in[i2-1]+m_Degrees_in[i4-1]);
        // Term 2
        term2 = m_Degrees_out[i1-1] * m_Degrees_in[i4-1] + m_Degrees_out[i3-1] * m_Degrees_in[i2-1]
      - m_Degrees_out[i1-1] * m_Degrees_in[i2-1] - m_Degrees_out[i3-1] * m_Degrees_in[i4-1];
        // Term 3 - the tr(c c^T c) term
        term3 += m_Connection[(i2-1) * m_Nodes + i4-1] + m_Connection[(i4-1) * m_Nodes + i2-1]
               + m_Connection[(i1-1) * m_Nodes + i3-1] + m_Connection[(i3-1) * m_Nodes + i1-1];
        term3 *= -2;
        term3 += CCTC_TrianglesBasedOn(i1, i4) + CCTC_TrianglesBasedOn(i3, i2)
               - CCTC_TrianglesBasedOn(i1, i2) - CCTC_TrianglesBasedOn(i3, i4);
        // Term 5 - the trace c^2 term
        term5 += m_Connection[(i1-1) + (i4-1) * m_Nodes] + m_Connection[(i3-1) + (i2-1) * m_Nodes]
               - m_Connection[(i1-1) + (i2-1) * m_Nodes] - m_Connection[(i3-1) + (i4-1) * m_Nodes];
        return term1 - term2 + term3 + term5; }
    if(swapType == Triangle) {
        int j1, j2, j3;
        int n_sqr_after_trig(0);
        int term1_3(0), term2_3(0), term3_3(0);
        if(i2 == i3) {
            j1 = i1; j2 = i2; j3 = i4; }
        else {
            if(i4 == i1) {
                j1 = i3; j2 = i4; j3 = i2; }
            else {
                cout << "I'm trying to calculate the incremental mobility, but I'm really confused which way my triangle goes";
                return 0; } }
        for(int cases = 0; cases < 6; ++cases) {
int alpha, beta, indicator; // save repeating code, I'll keep reassigning the value of these variable
            if(cases==0) { alpha = j1; beta = j2; indicator = -1; }
            if(cases==1) { alpha = j2; beta = j1; indicator =  1; }
            if(cases==2) { alpha = j2; beta = j3; indicator = -1; }
            if(cases==3) { alpha = j3; beta = j2; indicator =  1; }
            if(cases==4) { alpha = j3; beta = j1; indicator = -1; }
            if(cases==5) { alpha = j1; beta = j3; indicator =  1; }
            int temp = 0;
            for(unsigned int i =0; i<this->m_Nodes; i++) // for each stage I need to sum c_{alpha i} c_ {i \beta}
                temp += AreNeighbours(alpha, i+1) * AreNeighbours( i+1, beta);
            n_sqr_after_trig += temp * indicator;
            term1_3 += temp * indicator;
            temp = 0;
            for(unsigned int i = 0; i < m_Edges; ++i) {   // 3 * sum c_{km} c_ {k beta} c_ {alpha m }
                CCoordinate link = m_Neighbours[i];
                int k = link.x(), m = link.y();
                temp += AreNeighbours(k, beta)*AreNeighbours(alpha, m); }
            n_sqr_after_trig += temp * indicator *  2;
```

```
            term1_3 += temp * indicator * 2;
            n_sqr_after_trig += m_Degrees_out[beta-1] * m_Degrees_in[alpha-1] * indicator;
            term2_3 += m_Degrees_out[beta-1] * m_Degrees_in[alpha-1] * indicator; }
        n_sqr_after_trig += 3;
        return n_sqr_after_trig; }
    return 0; }

int CConnectionMatrix::CCTC_TrianglesBasedOn(int alpha, int beta) const {
    int N = m_Nodes;
    int sum = 0;
    for(int i = 0; i < N; ++i) {
        sum += m_Connection[(alpha - 1)*N + i] * m_Connection[i*N +(beta-1)];
        sum += m_Connection[(alpha - 1)*N + i] * m_Connection[(beta-1)*N +i];
        sum += m_Connection[i*N + (alpha - 1)] * m_Connection[i*N +(beta-1)]; }
    return sum; }

int CConnectionMatrix::IncrementalMobilityTermsTriangle(int i1, int i2, int i3, int i4, SwapType swapType) const {
    if(swapType == Square) {
        int delta_trig = 0;
        int indicator = 0;   /// I need an indicator if bond is created or destroyed
        int alpha, beta;
        for(int counter = 0; counter < 4; counter ++) { // 4 cases to sum over
            if(counter == 0) {beta = i3; alpha = i2; indicator =  1; }
            if(counter == 1) {beta = i1; alpha = i2; indicator = -1; }
            if(counter == 2) {beta = i1; alpha = i4; indicator =  1; }
            if(counter == 3) {beta = i3; alpha = i4; indicator = -1; }
            if(counter >= 4)
                throw "My IncrementalMobilityTermsTriangle subrOutine is cycling Out of control!";
            int sum;
            for(unsigned int j =1; j <= m_Nodes; ++j) {
                sum = 0;
                if(AreNeighbours(alpha,j))
                    if(AreNeighbours(j, beta))
                        sum += 1 + AreNeighbours(j, alpha) * AreNeighbours(beta, j) + AreNeighbours(j, alpha) * AreNeighbours(alpha, beta)
                             + AreNeighbours(alpha, beta) * AreNeighbours(beta, j) - AreNeighbours(alpha, beta) - AreNeighbours(j, alpha)
                             - AreNeighbours(beta, j) - AreNeighbours(j, alpha) * AreNeighbours(alpha, beta) * AreNeighbours(beta, j);
                if(AreNeighbours(beta,j))
                    if(AreNeighbours(j, alpha))
                        sum += AreNeighbours(j, beta) * AreNeighbours(alpha,beta) + AreNeighbours(alpha, beta) * AreNeighbours(alpha,j)
                             - AreNeighbours(alpha, beta) - AreNeighbours(j, beta) * AreNeighbours(alpha, beta) * AreNeighbours(alpha,j);
                delta_trig += indicator*sum; } }
        int sum = 0;
        if(AreNeighbours(i2, i4) == true && AreNeighbours(i4, i2) == false) {
            sum += (int) AreNeighbours(i4, i1) * (1 - AreNeighbours(i2, i1));
            sum += (int) AreNeighbours(i4, i3) * (1 - AreNeighbours(i2, i3)); }
        if(AreNeighbours(i1, i3) == true && AreNeighbours(i3, i1) == false) {
            sum += (int) AreNeighbours(i4, i1) * (1 - AreNeighbours(i4, i3));
            sum += (int) AreNeighbours(i2, i1) * (1 - AreNeighbours(i2, i3)); }
        if(AreNeighbours(i4, i2) == true && AreNeighbours(i2, i4) == false) {
            sum += (int) AreNeighbours(i2, i3) * (1 - AreNeighbours(i4, i3));
            sum += (int) AreNeighbours(i2, i1) * (1 - AreNeighbours(i4, i1)); }
        if(AreNeighbours(i3, i1) == true && AreNeighbours(i1, i3) == false) {
            sum += (int) AreNeighbours(i2, i3) * (1 - AreNeighbours(i2, i1));
            sum += (int) AreNeighbours(i4, i3) * (1 - AreNeighbours(i4, i1)); }
        delta_trig += sum;
        return delta_trig; }
    if(swapType == Triangle) {
        int j1, j2, j3;
        int n_trig_after_trig(0);
        if(i2 == i3) {
            j1 = i1; j2 = i2; j3 = i4; }
```

```
        else {
            if(i4 == i1) {
                j1 = i3; j2 = i4; j3 = i2; }
            else
                throw "I'm trying to calculate the incremental mobility, but I'm really confused which way my triangle goes"; }
        for(int cases = 0; cases < 6; ++cases) {
            int alpha, beta, indicator;
            if(cases == 0) {alpha = j1; beta = j2; indicator = -1; }
            if(cases == 1) {alpha = j2; beta = j1; indicator =  1; }
            if(cases == 2) {alpha = j2; beta = j3; indicator = -1; }
            if(cases == 3) {alpha = j3; beta = j2; indicator =  1; }
            if(cases == 4) {alpha = j3; beta = j1; indicator = -1; }
            if(cases == 5) {alpha = j1; beta = j3; indicator =  1; }
            for(unsigned int a = 0; a < m_Degrees_in[alpha-1]; ++a)
                for(unsigned int b = 0; b < m_Degrees_out[beta-1]; ++b)
                    if(GetNeighbours(alpha, In)[a] == GetNeighbours(beta, Out)[b])
                        if(AreNeighbours(alpha, GetNeighbours(alpha, In)[a]) == false && AreNeighbours(GetNeighbours(beta, Out)[b], beta) == false)
                            n_trig_after_trig += indicator; }
        n_trig_after_trig += 3;
        return n_trig_after_trig; }
    return 0; }

void CConnectionMatrix::MobilityTermSquare() {
    int n_sqr = 0, number_of_links = m_Edges;
    for(int x = 0; x < number_of_links; ++x)
        for(int y = 0; y< number_of_links; ++y)
            if(x != y) {
                int x1 = m_Neighbours[x].x(), y1 = m_Neighbours[x].y();
                int x2 = m_Neighbours[y].x(), y2 = m_Neighbours[y].y();
                if(AreNeighbours(x1,y2) == false && AreNeighbours(x2, y1) == false && y1 !=x2 && x1 != y2 && x1 != x2 && y1 != y2)
                    n_sqr ++; }
    n_sqr /= 2;
    m_Squares = n_sqr; }

void CConnectionMatrix::MobilityTermTriangle() {
    int n_trig = 0;
    for(unsigned int x = 1; x <= m_Nodes; ++x)
        for(unsigned int y = 1; y <= m_Nodes; ++y)
            if(AreNeighbours(x,y) * (1 - AreNeighbours(y,x)) == 1) {
                vector<int> y_Out = GetNeighbours(y, Out);
                vector<int> x_in = GetNeighbours(x, In);
                for(unsigned int a = 0; a < y_Out.size(); a++)
                    for(unsigned int b = 0; b < x_in.size(); b++)
                        if(y_Out[a] == x_in[b] && AreNeighbours(x,y_Out[a]) == false && AreNeighbours(y_Out[a],y) == false)
                            ++n_trig; }
    n_trig = n_trig/3;
    m_Triangles = n_trig; }

void CConnectionMatrix::InitialiseDirectedMobilityTerms() {
    MobilityTermSquare(); MobilityTermTriangle(); }

bool CConnectionMatrix::AcceptRejectDirected(const CCoordinate& link1, const CCoordinate& link2, SwapType swapType) {
    bool decision = false;
    double n_c = m_Squares + m_Triangles;       // get the old mobility term n_c=n_square+n_triangle
    /// get the new mobility term n_cnew = n_c + increment square + increment triangle
    int delta_square = IncrementalMobilityTermsSquare(link1.x(), link1.y(),link2.x(),  link2.y(), swapType);
    int delta_triangle = IncrementalMobilityTermsTriangle(link1.x(), link1.y(),link2.x(),  link2.y(), swapType);
    double n_cnew = n_c + delta_square + delta_triangle;
    double Acceptance_Probability = n_c / (n_c + n_cnew);         // my acceptance probability is 1/[1 + n_cnew/n_c]
    MTRand mtrand1;
    double randNum = mtrand1();
```

```
    if( randNum < Acceptance_Probability)
        decision = true;
    if(decision == true) {
        m_Squares += delta_square;
        m_Triangles += delta_triangle; }
    return decision; }
```

F.1.3 Main.h and Main.cpp

```
void FunctionFile(const char* filename, const int choice) {
    CConnectionMatrix C(filename);
    RandomiseRunner( C, filename,choice,1,false); }

void RandomiseRunner(CConnectionMatrix& C, const char* filename, const int choice, const int numSims, const bool statistics) {
    CConnectionMatrix COriginal = C;
    if(choice==13) {
        C.InitialiseDirectedMobilityTerms();
        int numberofswitches = C.NumberSwitches();
        FILE *file;
        fopen_s( &file, "C:/data/directed_randomiser/mobility_terms_correct.txt", "a");
        double mob_total1000_square = 0.0;
        double mob_total1000_triangle = 0.0;
        int j = 0;
        bool decision;
        int obsinterval = max(1,numberofswitches/100);
        fprintf(file," I'm attempting %d switches outputting every %d steps \n",numberofswitches, obsinterval);
        for(int i = 0; i < numberofswitches; i++) {
            SwapType swapType = NoSwap;
            do {
                decision = false;
                // pick 2 links & identify what kind of swap they do
                CCoordinate& link1 = C.RandomLink();
                CCoordinate& link2 = C.RandomUnconnectedLink(link1, swapType);
                if(swapType == Square) {
                    /// This subroutine is keeping track of my average mobility over the observation window
                    if(i % obsinterval == 0 ) {
                        fprintf(file," %f \t %f \n",mob_total1000_square/obsinterval, mob_total1000_triangle/obsinterval);
                        mob_total1000_square = C.NumberSquares();
                        mob_total1000_triangle = C.NumberTriangles(); }
                    else {
                        mob_total1000_square += C.NumberSquares();
                        mob_total1000_triangle += C.NumberTriangles(); }
                    ///choose whether to accept or reject the move
                    decision = C.AcceptRejectDirected(link1, link2, swapType); }
                if(swapType == Triangle) {
                    // I'm oversampling triangle moves, so discard 2/3 of samples immediately
                    double randNum = static_cast<double>(rand()) / static_cast<double>(RAND_MAX);
                    double correction = 1/3;
                    if(randNum<correction) {
                        if(i % obsinterval == 0 ) {
                            fprintf(file," %f \t %f \n",mob_total1000_square/obsinterval, mob_total1000_triangle/obsinterval);
                            mob_total1000_square = C.NumberSquares();
                            mob_total1000_triangle = C.NumberTriangles(); }
                        else {
                            mob_total1000_square += C.NumberSquares();
                            mob_total1000_triangle += C.NumberTriangles(); }
```

```
                    decision = C.AcceptRejectDirected(link1, link2, swapType); }
                else {
                    swapType = NoSwap; } }
            /// If I accept, update my object and add the incremental mobility terms to the old mobility term
            if(decision == true) {
                if(swapType == Square)
                    C.SwitchPair(link1, link2);
                if(swapType == Triangle)
                    C.ReverseTriangle(link1, link2);
                if(swapType != Square && swapType != Triangle) {
                    cout << "I've forgotten what kind of swap I'm meant to be doing!!";
                    break; } } }
        while(swapType == NoSwap); }
    fclose(file); }
if(choice == 14) {
    int numberofswitches = C.NumberSwitches();
    C.SwitchPairs(numberofswitches); }
C.SaveTab(choice);
FILE *file2;
fopen_s( &file2, "C:/data/directed_randomiser/log.txt", "a");
fprintf(file2,"Thank you for using directed graph randomiser\n");
fprintf(file2,"For your information, you have been working with:" );
fprintf(file2,filename);
fprintf(file2,"\n");
if(choice == 13)
    fprintf(file2,"You asked to randomise using the correct mobility terms\n");
else {
    if(choice ==14)
        fprintf(file2,"You asked to randomise accepting all proposed moves\n");
    else {
        if(choice ==15)
            fprintf(file2,"You asked to randomise using correct mobility term, and targetting the (original) Pi \n");
        else
            fprintf(file2,"I got confused about how your wanted me to randomise\n"); } }
double Hamming = C.HammingDistance(COriginal);
fprintf(file2,"I achieved a hamming distance of: %f\n", Hamming);
fprintf(file2,"I have saved separate files in this directory with the trajectory of my mobility terms and my final network.\n");
fclose(file2); }
```

F.1.4 Extension for Target Pi

We insert the following snippet into the initialisation function (shown below for the undirected case).

```
class CConnectionMatrix {
    [ ... ]
    int m_3Loops;
    int m_4Loops;
    int m_3LoopsDelta;
    int m_4LoopsDelta;
    double m_WeightNa;
    double m_WeightNb;
    double m_WeightNbPrev;          // Assign to the largest double initially
    double m_Probability;
    int m_WeightedAvgNeighbours;
    int m_WeightedAvgNeighboursAdj;
    vector<int> m_CorrelationCounter;
```

```
    double GetCorrelation(int, int);
    void DeltaLoops(CCoordinate&, CCoordinate&) };

void CConnectionMatrix::Initialise() {
    [ ... ]
    for(unsigned int i = 0; i < m_Nodes; ++i) {
        for(unsigned int j = 0; j < tempNeighbours[i].size(); ++j)
            for(unsigned int k = 0; k < tempNeighbours[i].size(); ++k) {
                if (AreNeighbours(tempNeighbours[i][j], tempNeighbours[i][k]))
                    ++m_3Loops;
                for(unsigned int m = 0; m < tempNeighbours[tempNeighbours[i][k]-1].size(); ++m)
                    if (AreNeighbours(tempNeighbours[i][j], tempNeighbours[tempNeighbours[i][k]-1][m]))
                        ++m_4Loops; } }
    m_3Loops /= 6;
    m_4Loops /= 8;
    for (unsigned int i = 0; i < m_Nodes; ++i) {
        int dummy(0);
        for (unsigned int j = 0; j < m_Degrees[i]; ++j)
            dummy += m_Degrees[tempNeighbours[i][j]-1];
        dummy *= m_Degrees[i];
        m_WeightedAvgNeighbours += dummy; }
    m_WeightNa = 0.25 * m_DegreeAvg * (1.0 * m_DegreeAvg * (double) m_Nodes) - 0.5 * m_DegreeVar;
    m_WeightNa *= (double) m_Nodes;
    m_CorrelationCounter.resize(m_MaxDegree * m_MaxDegree, 0);
    for (unsigned int i = 0; i < m_Nodes; ++i)
        for (unsigned int j = 0; j < i; ++j)
            if (AreNeighbours(i + 1, j + 1)) {
                m_CorrelationCounter[(m_Degrees[i] - 1) + m_MaxDegree * (m_Degrees[j] - 1)] += 1 ;
                m_CorrelationCounter[(m_Degrees[j] - 1) + m_MaxDegree * (m_Degrees[i] - 1)] += 1 ; } }

double CConnectionMatrix::GetCorrelation(int Node1, int Node2) {
    double invNodes = 1.0 / double(m_Nodes);
    double factor = m_DegreeAvg * (1.0 - invNodes) * invNodes;
    int deg1 = m_Degrees[Node1 - 1], deg2 = m_Degrees[Node2 - 1];
    if(deg1 * deg2 != 0) {
        double countValue = double(m_CorrelationCounter[(deg1 - 1) + m_MaxDegree * (deg2 - 1)]);
        double degdegcorrelation = 0.0;
        if (countValue != 0.0) {
            if (deg1 == deg2)
                degdegcorrelation = (countValue * factor / (  deg1 * deg2 * m_DegreePDF[deg1-1] * (m_DegreePDF[deg2-1] - invNodes) ) );
            else
                degdegcorrelation = (countValue * factor / (  deg1 * deg2 * m_DegreePDF[deg1-1] * m_DegreePDF[deg2-1] ) ); }
        return ( m_Nodes * m_DegreeAvg ) /  (deg1 * deg2 * degdegcorrelation) - 1.0; }
    else
        return 0.0; }

void CConnectionMatrix::DeltaLoops(CCoordinate& link1, CCoordinate& link2) {
    int i = link1.x(), j = link2.x(), k = link1.y(), l = link2.y();
    m_3LoopsDelta = 0; m_4LoopsDelta = 0;
vector<int> neighboursK = GetNeighbours(k), neighboursI = GetNeighbours(i),
                neighboursL = GetNeighbours(l),neighboursJ = GetNeighbours(j);
    for(unsigned int nk = 0; nk < m_Degrees[k-1]; ++nk) {
        int neighbour1 = neighboursK[nk];
        bool notDuplicate1 = (neighbour1 != i) && (neighbour1 != j) && (neighbour1 != k) && (neighbour1 != l);
        if (AreNeighbours(neighbour1, l) && notDuplicate1) {
            ++m_3LoopsDelta++;
            if(AreNeighbours(k, j))
                --m_4LoopsDelta;
            if(AreNeighbours(i, l))
                --m_4LoopsDelta; }
        if(AreNeighbours(neighbour1, i) && notDuplicate1) {
```

```
            --m_3LoopsDelta;
            if(AreNeighbours(k, j))
                ++m_4LoopsDelta;
            if(AreNeighbours(i, l))
                ++m_4LoopsDelta; }
        for(unsigned int ni = 0; ni < m_Degrees[i-1]; ++ni) {
            int neighbour2 = neighboursI[ni];
            bool notDuplicate2 = (neighbour2 != i) && (neighbour2 != j) && (neighbour2 != k) && (neighbour2 != l);
            if(AreNeighbours(neighbour1, neighbour2) && notDuplicate1 && notDuplicate2)
                --m_4LoopsDelta; }
        for(unsigned int nl = 0; nl < m_Degrees[l-1]; ++nl) {
            int neighbour2 = neighboursL[nl];
            bool notDuplicate2 = (neighbour2 != i) && (neighbour2 != j) && (neighbour2 != k) && (neighbour2 != l);
            if(AreNeighbours(neighbour1, neighbour2) && notDuplicate1 && notDuplicate2)
                ++m_4LoopsDelta; } }
    for(unsigned int nj = 0; nj < m_Degrees[j-1]; ++nj) {
        int neighbour1 = neighboursJ[nj];
        bool notDuplicate1 = (neighbour1 != i) && (neighbour1 != j) && (neighbour1 != k) && (neighbour1 != l);
        if (AreNeighbours(neighbour1, i) && notDuplicate1) {
            ++m_3LoopsDelta;
            if(AreNeighbours(k, j))
                --m_4LoopsDelta;
            if(AreNeighbours(i, l))
                --m_4LoopsDelta; }
        if(AreNeighbours(neighbour1, l) && notDuplicate1) {
            --m_3LoopsDelta;
            if(AreNeighbours(k, j))
                ++m_4LoopsDelta;
            if(AreNeighbours(i, l))
                ++m_4LoopsDelta; }
        for(unsigned int nl = 0; nl < m_Degrees[l-1]; ++nl) {
            int neighbour2 = neighboursL[nl];
            bool notDuplicate2 = (neighbour2 != i) && (neighbour2 != j) && (neighbour2 != k) && (neighbour2 != l);
            if (AreNeighbours(neighbour1, neighbour2) && notDuplicate1 && notDuplicate2)
                --m_4LoopsDelta; }
        for(unsigned int ni = 0; ni < m_Degrees[i-1]; ++ni) {
            int neighbour2 = neighboursI[ni];
            bool notDuplicate2 = (neighbour2 != i) && (neighbour2 != j) && (neighbour2 != k) && (neighbour2 != l);
            if (AreNeighbours(neighbour1, neighbour2) && notDuplicate1 && notDuplicate2)
                ++m_4LoopsDelta; } }
    m_WeightNb = 0.25 * 8.0 * (m_4Loops + m_4LoopsDelta) + 0.5 * 6.0 * (m_3Loops + m_3LoopsDelta);
    m_WeightedAvgNeighboursAdj = m_Degrees[i-1] * (m_Degrees[j-1] - m_Degrees[k-1]) +
                                 m_Degrees[j-1] * (m_Degrees[i-1] - m_Degrees[l-1]) +
                                 m_Degrees[k-1] * (m_Degrees[l-1] - m_Degrees[i-1]) +
                                 m_Degrees[l-1] * (m_Degrees[k-1] - m_Degrees[j-1]);
    m_WeightNb -= 0.5 * (double) (m_WeightedAvgNeighbours + m_WeightedAvgNeighboursAdj); }



void CConnectionMatrix::AcceptReject(CCoordinate& link1, CCoordinate& link2, int& counter) {
    int i = link1.x(), j = link2.x(), k = link1.y(), l = link2.y();
    ++counter;
    double ij = GetCorrelation(i, j), kl = GetCorrelation(k, l), ik = GetCorrelation(i, k), lj = GetCorrelation(l, j);
    double probability = (ij * kl) / (ik * lj);
    m_Probability = (m_WeightNa + m_WeightNbPrev) / (m_WeightNa + m_WeightNbPrev + (m_WeightNa + m_WeightNb) * probability);
    double randNum = rand() / ((double) RAND_MAX + 1);
    if( randNum < m_Probability ) {
        m_WeightNbPrev = m_WeightNb;
        m_3Loops += m_3LoopsDelta;
        m_4Loops += m_4LoopsDelta;
        m_3LoopsDelta = 0;
```

```
        m_4LoopsDelta = 0;
        m_WeightedAvgNeighbours += m_WeightedAvgNeighboursAdj;
        SwitchPair( link1, link2); } }
```

F.1.5 MYSQL manipulations

The gene regulation data for the calculations described in section 7.1 was provided in a MySQL database. To make the most direct use of the strengths of C++ and MySQL, this code was initially implemented via a MySQL connector. This meant that the data was arranged into the most convenient form within MySQL, and then the entropy is calculated within C++. This carries a speed penalty compared to doing all the operations within C++, but being able to directly manipulate interface with MySQL (and pre-process data within MySQL) makes usage more flexible and straightforward.

```
void MYSQL_manipulations(MYSQL *hnd) {
    mysql_query(hnd, "DROP TABLES node_list");
    string tabe = (string) TABLE_OF_INTEREST;
    string startofQ = "CREATE VIEW nod AS (SELECT orf1 as reference_node FROM ";
    string midofQ = " GROUP BY orf1) UNION (SELECT orf2 as reference_node FROM ";
    string endofQ = " GROUP BY orf2)";
    string fullQ = startofQ + tabe + midofQ +tabe+ endofQ;
    mysql_query(hnd, fullQ.c_str());
    mysql_query(hnd, "CREATE TABLE node_list AS SELECT * FROM nod GROUP BY reference_node");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "DROP VIEW nod");
    mysql_query(hnd, "ALTER TABLE node_list ADD id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY");
    mysql_query(hnd, "DROP TABLES tabl, temp");
    startofQ = "CREATE TABLE temp AS SELECT id AS orf2_id, orf1, orf2 FROM  ";
    midofQ = " LEFT JOIN node_list ON node_list.reference_node=";
    endofQ = ".orf2";
    fullQ = startofQ + tabe + midofQ +tabe+ endofQ;
    mysql_query(hnd,fullQ.c_str());
    mysql_query(hnd, "DROP TABLES out_batch, in_batch, join1, join2, join3, temp2");
    mysql_query(hnd, "CREATE TABLE temp2 SELECT * FROM temp WHERE orf1 != orf2");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE tabl SELECT * FROM temp2 GROUP BY orf1, orf2");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE out_batch SELECT orf1 AS node_id, COUNT(*) AS outie FROM tabl GROUP BY orf1");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE in_batch SELECT orf2, COUNT(*) AS indie FROM tabl GROUP BY orf2");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE join1 SELECT reference_node, outie FROM out_batch RIGHT JOIN node_list ON
                      out_batch.node_id = node_list.reference_node");
```

```
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE join2 SELECT reference_node, indie FROM node_list LEFT JOIN in_batch ON
                     node_list.reference_node = in_batch.orf2");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "ALTER TABLE join2 CHANGE reference_node r1 VARCHAR(80)");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE join3 SELECT indie, outie, reference_node FROM join1 CROSS JOIN join2 ON
                     join1.reference_node = join2.r1");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "DROP TABLES w, null_removed");
    mysql_query(hnd, "CREATE TABLE bond_list SELECT orf1, orf2 FROM tabl");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE step_1 SELECT bond_list.orf1, bond_list.orf2, indie, outie FROM bond_list
                     LEFT JOIN join3 ON bond_list.orf2=join3.reference_node");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE pre_W SELECT join3.indie AS origin_indie,join3.outie AS origin_outie,
                     step_1.orf1 AS start_node, step_1.orf2 AS end_node, step_1.indie AS finish_indie,
                     step_1.outie AS finish_outie FROM join3 LEFT JOIN step_1 ON step_1.orf1=join3.reference_node");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "CREATE TABLE null_removed SELECT * FROM pre_W WHERE start_node IS NOT NULL AND end_node IS NOT NULL");
    error = mysql_error(hnd);
    printf(error);
    mysql_query(hnd, "DROP TABLES step_1, pre_W, bond_list");
    mysql_query(hnd, "CREATE TABLE w SELECT origin_indie, origin_outie, COUNT(*) AS countie, finish_indie,
                     finish_outie FROM null_removed GROUP BY origin_indie, origin_outie, finish_indie, finish_outie");
    error = mysql_error(hnd);
    printf(error); }
```

This is the subroutine which loads the data from MySQL (note the *dataHandle input into the function), and derives some basic properties. In principle the same technology could be used for undirected graphs, but this case is specifically for directed.

```
CConnectionMatrix::CConnectionMatrix(MYSQL *dataHandle) : m_InitStage(None) {
    MYSQL_RES *res = NULL;
    MYSQL_ROW row;
    char sql[1024];
    int N =0;
    sprintf(sql,"select count(*) from node_list");
    // Here I find the number of nodes
    if (!mysql_query(dataHandle,sql)) {
        res = mysql_use_result(dataHandle);
        if (res) {
            row = mysql_fetch_row(res);
            while (row) {
                N = atoi(row[0]);
                row = mysql_fetch_row(res); }
            mysql_free_result(res); }
        else
            fprintf(stderr,"Failed to use the result acquired!\n"); }
    else
        fprintf(stderr,"Failed to execute query. Ensure table is valid!\n");
```

```
m_Nodes = N;
// Resize the vector and set all elements to false
m_Connection.resize(m_Nodes * m_Nodes, false);
for (unsigned int n = 0; n < m_Nodes; n++) {
    string node_ref;
    *res = NULL;
    string reference_node;
    sprintf(sql,"select reference_node from node_list where id=%d", n+1);
    if (!mysql_query(dataHandle,sql)) {
        res = mysql_use_result(dataHandle);
        if (res) {
            row = mysql_fetch_row(res);
            while (row) {
                reference_node = row[0];
                row = mysql_fetch_row(res); }
            mysql_free_result(res); } }
    node_ref = reference_node;
    // Find neighbour sub
    *res=NULL; // result of querying for all rows in table
    string reference_node;
    sprintf(sql, "select orf2_id from tabl where orf1='%s'",  node_ref.c_str());
    /// Here I find the number of nodes
    if (!mysql_query(dataHandle,sql)) {
        res = mysql_use_result(dataHandle);
        if (res) {
            vector<string> neighbours_of_this_node;
            row = mysql_fetch_row(res);
            while (row) {
                int pos = atoi(row[0]);
                m_Connection[n*this->m_Nodes + (pos - 1)] = true;
                CCoordinate coord( n+1, pos);
                m_Neighbours.push_back(coord);
                row = mysql_fetch_row(res); }
            mysql_free_result(res); }
        else
            fprintf(stderr,"Failed to use the result acquired!\n"); }
    else
        fprintf(stderr,"Failed to execute query. Ensure table is valid!\n");
    // Update m_Edges to be the number of connections.
    m_Edges = m_Neighbours.size(); }
// NOW USE THE DATA TABLES TO LOAD THE FREQUENCY TABLE
// I WILL CERTAINLY BE PUNISHED HERE FOR SPEED, BUT MYSQL IS EASIER TO DEFINE DATA MANIPULATIONS IN
*res = NULL; // result of querying for all rows in table
string reference_node;
sprintf(sql, "select * from w");
if (!mysql_query(dataHandle,sql)) {
    res = mysql_use_result(dataHandle);
    if (res) {
        row = mysql_fetch_row(res);
        while (row) {
            /// read the data
            int i_IN, i_OUT, j_IN, j_OUT;
            if(row[0] != NULL)
                i_IN = atoi(row[0]);
            else
                i_IN = 0;
            if(row[1] != NULL)
                i_OUT = atoi(row[1]);
            else
                i_OUT = 0;
            int countie = atoi(row[2]);
```

```
            if(row[3] != NULL)
                j_IN = atoi(row[3]);
            else
                j_IN = 0;
            if(row[4] != NULL)
                j_OUT = atoi(row[4]);
            else
                j_OUT = 0;
            //  Populate m_n'_d' as a vector of co-ordinates giving start-degree-end-degree sequence for each bond
            CCoordinate coord_i(i_IN, i_OUT);
            m_neighbouring_degrees.push_back(coord_i);
            CCoordinate coord_j(j_IN, j_OUT);
            m_neighbouring_degrees.push_back(coord_j);
            m_neighbouring_degrees_frequency.push_back(countie);
            row = mysql_fetch_row(res); }
        mysql_free_result(res); }
    else
        fprintf(stderr,"Failed to use the result acquired!\n"); }
else
    fprintf(stderr,"Failed to execute query. Ensure table is valid!\n");
cout << "loading from file took" <<end_loader-start_loader << "\n";
m_InitStage = BASE; }
```

F.2 Protein complex network interface

```
class Node {
public:
    Node(){}
    Node(string n1, string n2) : m_n1(n1), m_n2(n2){}
    bool operator< (const Node&) const;
    friend bool operator== (const Node&, const Node&);
    friend bool operator!= (const Node&, const Node&);
    friend bool Touching(const Node&, const Node&, bool = false);
    friend bool AreTriangle(const Node&, const Node&, const Node&);
    friend bool AreSquare (const Node&, const Node&, const Node&, const Node&);
    string Node1() const {return m_n1;}
    string Node2() const {return m_n2;}
private:
    string m_n1, m_n2; };

bool Node::operator< (const Node& x) const {
    string tmp1 = m_n1 < m_n2 ? m_n1 + "_" + m_n2 : m_n2 + "_" + m_n1;
    string tmp2 = x.m_n1 < x.m_n2 ? x.m_n1 + "_" + x.m_n2 : x.m_n2 + "_" + x.m_n1;
    return tmp1 < tmp2; }

bool operator== (const Node& x, const Node& y) {
    bool tmp1 = (x.m_n1 == y.m_n1) && (x.m_n2 == y.m_n2);
    bool tmp2 = (x.m_n1 == y.m_n2) && (x.m_n2 == y.m_n1);
    return tmp1 || tmp2; }

bool operator!= (const Node& x, const Node& y) {
    return !(x == y); }

bool Touching(const Node& x, const Node& y, bool checkSame) {
    if (checkSame)
        if (x == y)
            return false;
    return (x.m_n1 == y.m_n2) || (x.m_n1 == y.m_n1) || (x.m_n2 == y.m_n1) || (x.m_n2 == y.m_n2); }

bool AreTriangle(const Node& x, const Node& y, const Node& z) {
    set<string> tmp;
    tmp.insert(x.m_n1); tmp.insert(x.m_n2); tmp.insert(y.m_n1);
    tmp.insert(y.m_n2); tmp.insert(z.m_n1); tmp.insert(z.m_n2);
    return tmp.size() == 3; }

bool AreSquare(const Node& w, const Node& x, const Node& y, const Node& z) {
vector<string> tmp(8);
tmp[0] = w.m_n1; tmp[1] = w.m_n2; tmp[2] = x.m_n1; tmp[3] = x.m_n2;
    tmp[4] = y.m_n1; tmp[5] = y.m_n2; tmp[6] = z.m_n1; tmp[7] = z.m_n2;
    vector<int> node_frequency(8);
    for(int s = 0; s < 8 ; ++s) {       // Count how many times each node appears in list
        for(int t = 0; t < 8 ; ++t)
            if(tmp[s] == tmp[t])
                ++node_frequency[s];
        if (node_frequency[s] == 2)
            node_frequency[s] = 1;
        else
            node_frequency[s] = 0; }
    bool indicator = true;
    for(int s = 0; s < 8 ; ++s)
        for(int t = s; t < 8; ++t)
            indicator = indicator && (node_frequency[t] == node_frequency[s]);
    return indicator; }
```

```
int main() {
    const char* filename("c:\\data\\Protein_Complex_Network\\protein_complex_data.txt");
    FILE* file;
    fopen_s(&file, filename, "r");
    if (file == NULL)
        throw "Error, file not loaded";
    vector<pair<string, string> > protein_complexes;    // Create the vector of pairs
    char tempCharArray[50];
    int i = 0;
    cout << "Is your data in pairs with complex on the left and the consitutent protein on the right?";
    string mystr;
    getline (cin, mystr);
    if (mystr == "yes" || mystr == "y") {
        bool element1 = true;
        while (!feof(file)) {                   // Loop until we reach the end of the file
            char tempChar;
            fscanf_s(file, "%c", &tempChar);    // Read in the character
            if (tempChar == '\t' || tempChar == '\n')
                tempChar = '\0';                // If the character is either a tab or newline, then null terminate
            tempCharArray[i] = tempChar;        // Write to the array
            ++i;
            if (tempChar == '\0') {             // Check for string termination
                if (element1) {                 // If this is an element1 the extend the vector and add to the first position
                    protein_complexes.resize(protein_complexes.size() + 1);
                    protein_complexes.back().first = tempCharArray; }
                else
                    protein_complexes.back().second = tempCharArray;
                i = 0;                          // Reset counters
                element1 = !(element1); } } }
    else {
        cout << "The other format is a table: complex name is on the left and subsequent columns list the constituent proteins";
        bool protein = false;
        char tempCharArray2[50];
        while (!feof(file)) {                   // Loop until we reach the end of the file
            char tempChar_complex;
            char tempChar_protein;
            do {
                fscanf_s(file, "%c", &tempChar_complex);
                if (tempChar_complex == '\t' || tempChar_complex == '\n')
                    tempChar_complex = '\0';    // If the character is either a tab or newline, then null terminate
                tempCharArray[i] = tempChar_complex;
                ++i;
                if (tempChar_complex == '\0')   // Check for string termination
                    protein = true; }
            while (protein == false);
            i = 0;
            do {
                fscanf_s(file, "%c", &tempChar_protein);
                if (tempChar_protein == '\t')
                    tempChar_protein = '\0';    // I need to null terminate
                tempCharArray2[i] = tempChar_protein;
                if (tempChar_protein == '\n')
                    tempCharArray2[i] = '\0';
                ++i;
                if (tempChar_protein == '\0') { // Check for string termination
                    i=0;
                    if(tempCharArray2[0] !='\0') {
                        protein_complexes.resize(protein_complexes.size() + 1);
                        protein_complexes.back().first = tempCharArray;
                        protein_complexes.back().second = tempCharArray2; } } }
            while(tempChar_protein != '\n' && i < 50);
```

```
            i = 0;              // Reset counters
            protein = false; } }
    set<string> unique_protein_complexes;       // A unique set of protein complexes
      for (unsigned int i = 0; i < protein_complexes.size(); ++i)
          if (protein_complexes[i].first != "")
              unique_protein_complexes.insert(protein_complexes[i].first);
      set<Node> linked_proteins; set<string> nodes;       // Step through all unique complexes and push back into vector
      for (set<string>::iterator it = unique_protein_complexes.begin(); it != unique_protein_complexes.end(); ++it) {
          string protein_complex = it->c_str();
          set<string> nodes2;
          for (unsigned int i = 0; i < protein_complexes.size(); ++i)     // Filter the protein_complexes vector
              if (protein_complexes[i].first == protein_complex) {
                  nodes2.insert(protein_complexes[i].second);
                  nodes.insert(protein_complexes[i].second); }
          for (set<string>::iterator it = nodes2.begin(); it != nodes2.end(); ++it)
              for (set<string>::iterator it2 = it; it2 != nodes2.end(); ++it2) {
                  if (it == it2)
                      continue;
                  else
                      linked_proteins.insert(Node(*it, *it2)); } }
      const char* outputname("c:\\data\\Protein_Complex_Network\\results_hprd.txt");
      FILE* output;
      fopen_s(&output, outputname, "w");
      fputs("This is the file of all linked proteins \n", output);
      for ( set<Node>::iterator it = linked_proteins.begin(); it != linked_proteins.end(); ++it) {
          fputs(it->Node1().c_str(), output);
          fputs("\t", output);
          fputs(it->Node2().c_str(), output);
          fputs("\n", output); }
      cout << '\n'<< linked_proteins.size() << '\n';
      fputs("\n\nAdding triangle details\n", output);
      int TRIANGLEcounter(0);     // Triangle count
      int SQUAREcounter(0);       // Square count
      for (set<Node>::iterator it = linked_proteins.begin(); it != linked_proteins.end(); ++it)
          for (set<Node>::iterator it2 = it; it2 != linked_proteins.end(); ++it2) {
              if (it2 == it)
                  continue;
              if (Touching(*it, *it2))
                  for (set<Node>::iterator it3 = it2; it3 != linked_proteins.end(); ++it3) {
                      if (it3 == it2)
                          continue;
                      if (Touching(*it3, *it2)) {
                          if (AreTriangle(*it, *it2, *it3))
                              ++TRIANGLEcounter;
                          if ( Touching(*it3, *it) == false)
                              for (set<Node>::iterator it4 = it3; it4 != linked_proteins.end(); ++it4) {
                                  if (it4 == it2 || it4 == it3  || it4 == it )
                                      continue;
                                  if (AreSquare(*it, *it2, *it3, *it4))
                                      ++SQUAREcounter; } }
                      else
                          if ( Touching(*it3, *it) )
                              for (set<Node>::iterator it4 = it3; it4 != linked_proteins.end(); ++it4) {
                                  if (it4 == it2 || it4 == it3  || it4 == it )
                                      continue;
                                  if (AreSquare(*it, *it2, *it3, *it4))
                                      ++SQUAREcounter; } } }
      char buffer[100];
      sprintf_s(buffer, "%d", TRIANGLEcounter);
      fputs(buffer, output);
      cout << "number of triangles" << TRIANGLEcounter << '\n';
```

```
fputs("\n\nAdding SQUARE details\n", output);
char buffer2[100];
sprintf_s(buffer2, "%d", SQUAREcounter);
fputs(buffer2, output);
cout << "Number of squares" << SQUAREcounter << '\n';
cin >> i;
return 0; }
```